Developer Guide

Document Version: Q1 2018 – 2018-03-09

PUBLIC

SAP SuccessFactors HCM Suite OData API: Developer Guide

About HCM Suite OData APIs

Content

1 What's New in this Guide

This document describes changes to this guide for the recent releases. If you have feedback, please send an email to SAPSuccessFactorsDocumentation@sap.com.

There have been no updates to this guide since Q4 2017.

Q4 2017

The following table summarizes changes to this guide for Q3 2017 release

What's New	Description	More Information
Update to Refreshing and Exporting the Metadata	Information about scenarios requiring manual refresh has been added.	Refreshing and Exporting the Metadata [page 60]
Update to Merging with Existing Data	Information about merging data with navigation properties has been added and the JSON request sample data has been enhanced.	Merging with Existing Data [page 37]

Q3 2017

The following table summarizes changes to this guide for Q3 2017 release

What's New	Description	More Information
SF API Audit Log	New topic about about the usability and feature enhancements to the SF API Audit Log.	SF API Audit Log [page 58]
OData API Audit Log	Topic updated with information about the usability and feature enhancements to the OData API Audit Log.	OData API Audit Log [page 58]
API Center	The API center is available in the Admin Center as long as the user has permission for at least one of the tools hosted on the API center.	API Center [page 56]

What's New	Description	More Information
Manage User API Options	This is not a new feature but is now available from the API Center. Please note that if you access it directly from the Admin Center, it is called *Manage User API Options*.	Manage User API Options [page 59]
OData IP Whitelisting	This is not a new feature but it is now available from the API Center. Please note that if you access it directly from the Admin Center, it is called *OData IP Whitelisting*	OData IP Whitelisting [page 60]

Q2 2017

The following table summarizes changes to this guide for Q2 2017 release

What's New	Description	More Information
$batch: Upsert and changeset behavior	In contrast to the standard OData API $batch processing specification, a partial upsert failure reverts the transaction for the current changeset.	$batch: Upsert and changeset behavior [page 41]
Upsert parameter: strictTransactionIsolate	Describes how setting the value of this parameter facilitates changeset rollback in certain use cases.	Upsert parameter: strictTransactionIsolate [page 40]
OData API Data Dictionary	Topic updated with information about the Usability and feature enhancements to the OData API Data Dictionary	OData API Data Dictionary [page 56]
API Center	Introducing the one stop shop for your API tools	API Center [page 56]
The $orderby Keyword	$orderby only orders the top level entity and not child entities; in a 1: many navigation properties, the order can be unpredicable.	The $orderby Keyword [page 22]

Q1 2017: No updates

Q4 2016

The following table summarizes changes to this guide for Q4 2016 release

What's New	Description	More Information
Table containing endpoint urls updated.	More detailed information about the DC location added.	About HCM Suite OData APIs [page 7]

Q3 2016

The following table summarizes changes to this guide for Q3 2016 release

What's New	Description	More Information
August 5		
The topic *$filter Keyword* has been up-dated.	In this topic, the section *Customized Op-erators* has been updated with the follow-ing information "Due to the Oracle maxi-mum number of expressions (1000) in a list (ORA-01795), OData API does not ac-cept an IN clause exceeding more than 1000 expressions in the $filter."	The $filter Keyword [page 24]

Related Information

SF API Audit Log [page 58]

2 About HCM Suite OData APIs

The Open Data Protocol (OData) is a standardized protocol for creating and consuming data APIs. OData builds on core protocols like HTTP, and commonly accepted methodologies like REST. The result is a uniform way to expose full-featured data APIs. OData provides both a standard for how to represent your data and a metadata method to describe the structure of your data, and the operations available in your API. SuccessFactors OData API service is based on OData V2.0.

The HCM Suite OData API is SuccessFactors Web Services API based on OData protocol intended to enable access to data in the SuccessFactors system. The API is data oriented. This API provides methods for CRUD style access (Create, Read, Update and Delete). The API is best used for frequent or real time requests for small amounts of data. Large data requests are better handled by batch FTP processes. This OData API is used to configure entities. Each SuccessFactors module can be accessed using its own set of entities.

Enabling the HCM Suite OData API

The OData API switch in provisioning is set to enabled by default. The OData API feature is available by default, unless you manually turn it off in provisioning.

API Endpoint URLs

Your endpoint URLs for accessing the OData APIs depend on the data center hosting your SuccessFactors instance. Your SuccessFactors support representative can tell you the data center location to use for your instance.

Recommendation

We recommend using OAuth instead of basic authentication for accessing customer systems. Available in scenarios in which the Event Connector is used to integrate Platform systems with a customer system and where the endpoint is a SOAP API.

Below are the URLs for each of the SuccessFactors data centers:

Location	Environment	Endpoint URL (ODataServiceUrl)
Europe, The Netherlands, Amsterdam	DC2 Production	`https://api2.successfactors.eu/odata/v2/`
Europe, The Netherlands, Amsterdam	DC2 SalesDemo	`https://apisalesdemo2.successfactors.eu/odata/v2/`
Europe, The Netherlands, Amsterdam	DC2 Preview	`https://api2preview.sapsf.eu/odata/v2/`

Location	Environment	Endpoint URL (ODataServiceUrl)
USA, Arizona, Chandler	DC4 Production	`https://api4.successfactors.com/odata/v2/`
USA, Arizona, Chandler	DC4 SalesDemo	`https://apisalesdemo4.successfactors.com/odata/v2/`
USA, Arizona, Chandler	DC4 Preview	`https://api4preview.sapsf.com/odata/v2/`
n.a	DC5 Production	`https://api5.successfactors.eu/odata/v2/`
USA, Virginia, Ashburn	DC8 Production	`https://api8.successfactors.com/odata/v2/`
USA, Virginia, Ashburn	DC8 SalesDemo	`https://apisalesdemo8.successfactors.com/odata/v2/`
USA, Virginia, Ashburn	DC8 Preview	`https://api8preview.sapsf.com/odata/v2/`
Australia, Sydney	DC10 Production	`https://api10.successfactors.com/odata/v2/`
Australia, Sydney	DC10 Preview	`https://api10preview.sapsf.com/odata/v2/`
Rot, Germany	DC12 Production	`https://api012.successfactors.eu/odata/v2/`
Rot, Germany	DC12 Rot	`https://apirot.successfactors.eu/odata/v2/`
Rot, Germany	Preview	`https://api12preview.sapsf.eu/odata/v2/`
China, Shanghai	DC15 Production	`https://api15.sapsf.cn/odata/v2/`
Germany, Biere	DC16 Production	`https://api16.sapsf.eu/odata/v2/`
Canada, Toronto	DC17 Preview	`https://api17preview.sapsf.com/odata/v2/`
Canada, Toronto	DC 17 Production	`https://api17.sapsf.com/odata/v2/`
Russia, Moscow	DC18 Preview	`https://api18preview.sapsf.com/odata/v2/`
Russia, Moscow	DC18 Production	`https://api18.sapsf.com/odata/v2/`

3 Authentication Types for OData API

The OData API provides two types of authentication: HTTP Basic Authentication (Basic Auth) and authentication using OAth 2.0. Basic Auth requires users to have admin access to the OData API and have valid accounts with usernames and passwords. OAuth 2.0 lets all users log in regardless of whether they are SSO users, so long as they have a valid token and the OAuth client is registered.

3.1 HTTP Basic Authorization

You must have a admin access to OData API to login using HTTP Basic Authentication (Basic Auth). For Basic Auth, the authorization header is constructed in the following way:

- Username, Company ID, and password are combined into a string as such: "username@company ID:password"
- The resulting string literal is then encoded using Base64.
- The authorization method ("Basic") followed by a space is then put before the encoded string. For example, Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

Note

When SSO is enabled, users cannot use Basic Auth. A partial workaround is available through a feature called partial org sso for App login. Users who have been assigned the permission *Admin Access to OData API* from the RBP system can log in using a password-based login, even if they are an SSO user. Note that Basic Auth should be used only by users who need administrative access for system-to-system data integration.

3.1.1 Granting Permission for an RBP System

Once OData is enabled, you must authorize access to the API for an Role-Based Performance (RBP) system to activate Basic Auth for a given user.

1. Click *Admin Tools* > *Manage Security* > *Manage Permission Roles*.
2. To select the role for which you need to add API access, . under <uicontrol>Permission settings</uicontrol> click <uicontrol>Permissions</uicontrol>.
3. Click *Administrator Permissions* > *Manage Integration Tools*, and select the *Admin access to OData API* checkbox.

3.1.2 Granting Permission For a User-based System

After OData is enabled, to activate Basic Auth for a given user you must authorize access to the API for a user-based system

1. Click *Administrative Privileges* > *Integration Tools*, and select *Admin access to OData API*.
2. On the *Managing Administrative Privilege* page, select the *Employee Export* and *Employee Import* checkboxes to grant those permissions to the given user.

3.1.3 Authenticating from a Browser

You can use a web browser based authentication to enable querying..

To use this feature, you need:

1. Input URL of query OData in address bar. e.g. `https://<hostname>/odata/v2/Attachment?$top=1`
2. Enter your username and password when prompted.

You will not need to re-enter this information for subsequent queries until the browser is closed. If the user name and password verification fails, you will need to open a new window and try again

Note

When you query in atom format in Firefox, the xml result will not be displayed, because Firefox regards atom format as RSS content by default. You can check the result in JSON format or view the source code to check the query result.

3.2 Setting Password Policy Exceptions For API Login

Your SuccessFactors HCM Suite administrator sets password policies for all users in the system, including the timing for password expirations. However, you may want to set different expiration times for passwords for intergrations that are built against a specially designed API user account. You can set exceptions to the password policy for your API user, and specify a different password expiration (maximum password age) policy. To maintain security, users who have password policy exceptions are required to have an IP address restriction to connect to the API . Password policy exceptions are set in *Admin Tools* > *Company Settings* > *Password & Login Policy Settings*. Select the link labeled "*Set API login exceptions...*" to add a custom password expiration. You must provide a username, and the desired maximum password age in days (-1 means the password will not expire, though we do not recommend doing that). You must also provide an IP address restrictions for this user.

3.3 Authentication using OAuth 2.0

OAuth 2.0 lets all users log in regardless of whether they are SSO users . If you are planning to use OAuth 2.0 for authentication, you will first need to register your OAuth client, and set up the permissions required for. this registration. Then you can register your OAuth client application.

3.3.1 Granting Permissions for an RBP System

From the admin menu *Manage Permission Roles*, select the desired role for which you want to add the permission. As a best practice, create role named "API Administrator" . Under the *Manage Integration Tools* link, select the *Manage OAuth2 Client Applications* checkbox.

After you have done this, you will see a link, *Manage OAuth2 Client Applications*under the *Company Settings* category in the new admin tools, and under *Integration Tools* in the older administration tools interface.

3.3.2 Granting Permissions for a User-Based System

From the Admin Menu click *Manage Security* › *Administrative Privileges*. For the user you are logged in as, look under *Integration Tools* and check the box under *Access to OAuth 2 Management*.

After you have done this, you will see a link under *Integration Tools* to where you can register your OAuth client.

3.3.3 Registering your OAuth Client Application

To register an OAuth client, log into your application instance with an administrator account. From the Admin menu click *Manage OAuth2 Client Applications* › *Register New Client Application*. After you register an OAuth client, any user of the registered client can connect to SuccessFactors HCM Suite using this method.

Enter the following parameters in the form that is displayed, and click*Register*:

VALUE	DESCRIPTION
Company	The name of your company. This value is pre-filled based on the instance of the company currently logged in.
Application Name	A unique name of your OAuth client.
Description (optional)	An optional description of your application.
Application URL	A unique URL of the page that the client wants to display to the end user. The page might contain more information about the client application. This is needed for 3-legged OAuth, however it is not currently supported.

VALUE	DESCRIPTION
X.509 Certificate	The certificate corresponding to the private and public key used in the OAuth 2.0 authentication process. In this flow, the SuccessFactors HCM Suite system will need the public key (the certificate) and the client application will have the private key. To register a client application, you will need to install the public key (aka certificate) in SuccessFactors HCM Suite. If you supply that certificate, you must use the RSA-SHA1 signature type for authenticating. As an optional feature, you can generate a public and private key pair with. the *Generate X.509 Certificate* button. If you do this, you must download the private key (or key pair) and install it into your client application. SuccessFactors HCM Suite will keep a copy of the private key.
Generate X.509 Certificate Button	A button that generates a X.509 certificate if the customer doesn't have one already. When clicked, a dialog box is displayed, in which the customer can enter the following information then click "Generate" to generate a self-signed certificate:
	Issued By : Value set to *SuccessFactors*
	Common Name: The name or IP address for which the certificate is valid.
	Organization: (optional) The entity to which the certificate is issued.
	Organization Unit: (optional) The organization unit of the entity to which the certificate is issued.
	Locality: (optional) Name of Locality of the entity to which the certificate is issued.
	State/Province:(optional) Name of State or Province of the entity to which the certificate is issued.
	Country: (optional) Name of Country of the entity to which the certificate is issued.
	Validity: The number of days for which you want the X.509 certificate to be valid.

If you have generated the X-509 Certificate, you must download the private key to use it in your client application to make token requests. The system saves the public key. You will need to regenerate the private key if you lose it.

i Note

You will need to save the Private Key before you register you client. Only the Public Key is available for viewing when the client is registered. You will have the API Key and Private Key available to you in the generated certificate.

3.3.4 Managing an existing OAuth2 Client Application

You can modify the X 509 certificate and the application description, for an existing client application by clicking the *Edit* button for the client application on the application list page. In the client application edit page that appears, you can make the modifications.

You can either replace the existing X.509 certificate, or regenerate the certificate by clicking the *Generate X.509 Certificate* button.

3.3.5 Enabling/Disabling an existing OAuth 2 Client

You can enable or disable an existing OAuth 2 client by clicking the Disable (Enable) button on the client button for the client application on the application list page

3.3.6 Authentication APIs for OAuth 2.0

The SF OData API supports 2-legged authentication for OAuth 2.0. Users with valid accounts in a registered client can use the SuccessFactors OData API after receiving an OAuth 2.0 token to use in subsequent API calls.

3.3.6.1 Recommended Flow for using APIs

Following is the order in which the APIs are used.

API	DESCRIPTION AND USAGE
API for Generating SAML Assertion	Generate a signed SAML assertion.
API for Requesting a User Token Using SAML Assertion	Request a Token. A signed SAML Assertion identifying a SAML IDP and the user must be provided because a request parameter acts as a SAML service provider for the client's SAML IDP.
API for Validating Generated Token	Validate the generated token.
Access OData API	Access OData API with the generated Bearer authorization token.

3.3.6.2 API for Generating a SAML Assertion

The following are the APIs for generating a Security Assertion Markup Language (SAML) assertion for authentication.

RESOURCE	DESCRIPTION
REQUEST URL	oauth/idp
DESCRIPTION	Generates a SAML assertion.
REQUEST METHOD	POST
REQUEST POST	Content-Type: application/x-www-form-urlencoded

RESOURCE	DESCRIPTION
PARAMETERS	• client_id - OAuth client id. This is the API Key that was generated earlier. • user_id - NameId in the SAML assertion. • token_url – Recipient in SAML assertion. • private_key – used to sign SAML assertion. • use_email (optional) - accept "nameid-format:emailAddress" in saml assertion (b1511 and after)
RESPONSE HEADER	• 200: OK or • See OAuth 2.0 related Error Messages for more information.
RESPONSE BODY	Base64-encoded SAML assertion.

When "use_email=true" appears in the /oauth/idp request payload, the subject name id format in the generated saml assertion will now change from "nameid-format:unspecified" to "nameid-format:email". An email address has to be mapped to a unique BizX user. If such email can be mapped to more than one BizX userId, a HTTP 401 error ("Unable to map xxx@xxx.xxx to a valid BizX User ID") will be returned.

Related Information

OAuth 2.0 Related Error Messages [page 74]

3.3.6.3 API for Requesting a User Token Using a SAML Assertion

RESOURCE	DESCRIPTION
REQUEST URL	oauth/token
DESCRIPTION	Issues an OAuth 2.0 token for the user given by the supplied scope. An error is returned if the user is not found.
REQUEST METHOD	POST
REQUEST HEADER	Content-Type: application/x-www-form-urlencoded
PARAMETERS	• company_id - Id of the company • client_id - OAuth client id. • api_key: This is the API Key that was generated earlier. • assertion – Base64-coded SAML assertion. • grant_type – urn:ietf:params:oauth:grant-type:saml2-bearer
RESPONSE HEADER	• • 200: OK Or • See OAuth 2.0 related Error Messages for more informa-tion.

RESOURCE	DESCRIPTION
RESPONSE BODY	Contains the access token that you use in subsequent API calls: • An OAuth 2.0 token for the given user. This token must be used in the “Authorization” header in all subsequent requests. Described as below: ○ access_token: a valid OAuth access token. ○ token_type: "Bearer" (this is the only supported type). ○ expires_in: Remaining lifetime of the access token in seconds.

3.3.6.3.1 SAML Assertion Example Request

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2:Assertion ID="d3dbe3f1-867e-4f7e-85c0-7b9659350384"
    IssueInstant="2013-10-22T05:36:32.627Z" Version="2.0"
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/
2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <saml2:Issuer>ZTdmNmFmYTQ4YjI1OWEzZTBhZTI3ZmYxOGUxZg</saml2:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
            <ds:Reference URI="#d3dbe3f1-867e-4f7e-85c0-7b9659350384"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:Transforms xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:Transform
                        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"
                        xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"
                        xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
                    xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                <ds:DigestValue xmlns:ds="http://www.w3.org/2000/09/
xmldsig#">LQ382vMrHK4QV8QUMnimrdPCros=
                </ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            3x6JmWzqWCHl8ctRG9A/
bzKhJqULZROPfGFIal89ChGzqE8Jk7dgQauWa6867nJTCUAdDo401oGf
            vgijBtC8vkA5JolnebIN2ouJ4jJNK0WLxAkFqHndRi3Eo9qrx2H6n4KqIpD/X/
3hHfsecXDeW1WE
            CsVLBoVNxZ4xEqTrArnT6hOq5bY2jkfPEutRtS/VFW54OQPtLLiXi5nqAsLCGC/
VgZlqwClT5qKh
            5wwbJQYjcPKjYzqPNeF/zpjFMA7SsN
+ztnV0+CIxIX8SvUvpNVg9Zr3jLC4cOp2QYL9pWzzEYkuk
            LY3wI5uBZnOPZqVH0XitR8g7QWuigD/gPHX6zQ==
        </ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>

MIICDTCCAXagAwIBAgIETJj9LjANBgkqhkiG9w0BAQUFADBLMQswCQYDVQQGEwJVUzEbMBkGA1UE

ChMSU3VjY2Vzc2ZhY3RvcnMuY29tMQwwCgYDVQQLEwNPcHMxETAPBgNVBAMTCFNGIEFkbWluMB4X

DTEwMDkyMTE4NDUwMloXDTI1MDkxOTE4NDUwMlowSzELMAkGA1UEBhMCVVMxGzAZBgNVBAoTElN1

Y2Nlc3NmYWN0b3JzLmNvbTEMMAoGA1UECxMDT3BzMREwDwYDVQQDEwhTRiBBZG1pbjCBnzANBgkq

hkiG9w0BAQEFAAOBjQAwgYkCgYEArA9RLNnL9Pt6xynFfYfa8VXAXFDG9Y8xkgs3lhIOlsjqEYwb
                    SoghiqJIJvfYM45kx3aB7ZrN96tAR5uUupEsu/GcS6ACxhfruW+BY6uw8v6/
w2vXhBdfFjBoO+Ke

Lx4k3llleVgKsmNlf81okOXv1ree8wErfZ3ssnNxkuQgGB0CAwEAATANBgkqhkiG9w0BAQUFAAOB
                    gQBeBCSMFnY8TB6jtWoSP/lorBudhptgvO7/3r+l/QK0hdk6CVv
+VQmSilNPgWVgU9ktZGbNkZhw
                    IgwnqIQHAi6631ufkYQJB+48YUe1q/pv6EWaeIwGvcGYSXZp/E/
aGZPtceTIXFPfqOyHQoFtb0nq
                    MMFWoDhpXUHmlroyTc9sJg==
                </ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
    <saml2:Subject>
        <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">star</saml2:NameID>
        <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
            <saml2:SubjectConfirmationData
                NotOnOrAfter="2013-10-22T05:46:32.627Z" Recipient="http://
qacandsfapi/oauth/token" />
        </saml2:SubjectConfirmation>
    </saml2:Subject>
    <saml2:Conditions NotBefore="2013-10-22T05:26:32.627Z"
        NotOnOrAfter="2013-10-22T05:46:32.627Z">
        <saml2:AudienceRestriction>
            <saml2:Audience>www.successfactors.com</saml2:Audience>
        </saml2:AudienceRestriction>
    </saml2:Conditions>
    <saml2:AuthnStatement AuthnInstant="2013-10-22T05:36:32.627Z"
        SessionIndex="53e5bdee-29f1-49d9-a3e2-e509da46328c">
        <saml2:AuthnContext>
            <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:
2.0:ac:classes:PasswordProtectedTransport
            </saml2:AuthnContextClassRef>
        </saml2:AuthnContext>
    </saml2:AuthnStatement>
</saml2:Assertion>
```

3.3.6.4 API for Validating a Generated Token

RESOURCE	DESCRIPTION
REQUEST URL	oauth/validate
DESCRIPTION	Validates a given SAML token for the user given by the supplied scope. An error is returned if the user is not found.
REQUEST METHOD	GET

RESOURCE	DESCRIPTION
REQUEST HEADER	Authorization: Bearer XXXXXXXX
RESPONSE HEADER	• 200: OK Or • See OAuth 2.0 related Error Messages for more information.
RESPONSE BODY	Contains an access token, token type, and expiry information if the token is valid. Described as below:: • access_token: a valid OAuth access token • token_type: "Bearer" (this is the only supported type) • expires_in: Remaining lifetime of the access token in seconds.

3.4 Enabling Session Reuse

OData API is not stateful. By default, each OData call needs to go through authentication and authorization. This has a big effect on performance. To address this, you can enable session reuse as described in the following section.

1. Login providing authentication and authorization information.
2. The first login returns a header, `Set-Cookie` and a CSRF token named X-CSRF-Token.
3. Store the value of this cookie and token in the authorization header as a HTTP Cookie (header field names, `Cookie` and `X-CSRF-Token`) and use them in subsequent requests to tell the server to use the same logged in session.
4. If the cookie was passed correctly, then the login procedure would make use of the logged session, and you will see a message in the `server.log` that looks like this:`00:03:17,622 INFO [ODLoginFilter] Login using logged session:hHt73YMMuwopRNc1F9UD6g**.sfapi`

Note

1. While using this session, if login fails, the OData login API will fall back on Basic authentication. In other words, it is possible to have both this reuse session and Basic authentication information in a HTTP request headers- the OData API will try to login using the session reuse first.
2. Since these sessions use authorization and authentication only once, subsequent requests are not aware of any newer permission changes. Similar to a UI login, you will need to logout and login again to force reloading of latest permissions.

Sample Headers

1. The first OData login returns a cookie header and CSRF token:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
```

```
X-Powered-By: Servlet 2.4; JBoss-4.3.0.GA_CP09 (build:
SVNTag=JBPAPP_4_3_0_GA_CP09 date=201011090309)/JBossWeb-2.0
Set-Cookie: JSESSIONID=2oeUIX5Glw14EZXU7fDWMQ**.sfapi; Path=/
X-CSRF-Token: 97sHgXdBw6%2bCKGg3sWAahbT8ztI%3d
Location: https://localhost:443/odata/v2/User('admin')
DataServiceVersion: 1.0
SFODataServerTimeZone: America/Los_Angeles
Content-Type: application/atom+xml;charset=utf-8
Content-Length: 1518
Date: Thu, 14 Aug 2014 05:25:06 GMT
```

2. 2. Client takes this Cookie and CSRF token as request header:

```
Authorization: Basic YWRtaW5Ac2ZhcGk6cHdk
Content-Type: application/json; charset=utf-8
Cookie: JSESSIONID=2oeUIX5Glw14EZXU7fDWMQ**.sfapi
X-CSRF-Token: 97sHgXdBw6%2bCKGg3sWAahbT8ztI%3d
```

4 OData Operations

The OData API is a standardized protocol for accessing a database. The Entities in the API are described in the metadata document. Accessing the Entities is as simple as knowing the entity name, and then referencing the entity through an HTTP call. For example, all SuccessFactors HCM Suite solutions have a `User` Entity that represents a user account. To query all the users in a system, you type the URL https://<hostname>/odata/v2/User A query can be more complex by including filtering, paging, sorting, and joining of entities (known as expanding in OData).

i Note

OData protocol allows the service provider (SuccessFactors HCM Suite) to decide which operations are supported. In SuccessFactors HCM Suite, the supported operations vary by entity. For example, the system does not allow user accounts to be deleted. User accounts can be removed from use only by setting them as inactive. Therefore the `User` entity does not support the delete operation.

This document does not cover the details of each entity behavior for all SuccessFactors HCM Suite solutions. Details for each entity are provided in a separate reference document. That document provides details about operations, business meaning and business logic, and configuration details such as security and custom-field configuration.

4.1 Composing the OData URI

In OData, a URI has the following structure:

```
http://<hostname>/odata/v2/EntitySet[(keyPredicate)][/navPropSingle][/
NavPropCollection][/Complex][/Property]
```

See for descriptions of each element. :

The following table contains examples of a URIand a description of its composition:

SAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User`	• Identifies a `User` Collection. • Described by the Entity type named "User" in the meta-data document.
`http://<hostname>/odata/v2/User('1')`	• Identifies a single User entry with key value 1. • Described by the Entity type named "User" in the service metadata document.

SAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User('1')/username`	• Identifies the `username` property of the User entry with key value 1. • Described by the Property named username on the User entity type in the metadata document.
`http://<hostname>/odata/v2/User('1')/proxy`	• Identifies the collection of proxy associated with User entry with key value 1. • Described by the Navigation Property named `proxy` on the User entity type in the metadata document.
`http://<hostname>/odata/v2/User('1')/proxy/$count`	• Identifies the number of proxy Entries associated with User 1. • Described by the Navigation Property named "proxy" on the "User" entity type in the metadata document.
`http://<hostname>/odata/v2/User('1')/proxy('1')/hr/username`	• Identifies the username of the hr for proxy 1 which is associated with User 1. • Described by the Property named username on the `hr` entity type in the metadata document.
`http://<hostname>/odata/v2/User('1')/proxy('1')/hr/username/$value`	Same as the URI above, but identifies the raw value of the username property.

4.2 Date and Time

The SuccessFactors HCM Suite OData service supports the both `Edm.DateTime` and `Edm.DateTimeOffset`f ormats of the OData protocol. This support makes it easy for integration clients to convert date and time values to different time zones as needed.

4.2.1 Using the DataTime Format

For DateTime format, the input value is in UTC time, and it is stored and retrieved from the database in the same format.

Example of DateTime format

Atom Format

Input: If the input value in Atom format is 2014-4-22T18:10:10, the OData server treats the input string as UTC time, stores to the database directly.

Output: when requested, the date value is retrieved from the database and returnd to client with no date conversion.

JSON Format

Input: If the date is 2014-04-22T18:10:10, then convert it in milliseconds under UTC timezone. Time in milliseconds is 1398190210000, so the input date in json format is /Date(1398190210000)/ . When OData receives this value, it converts it to 2014-04-22T18:10:10, and stores it in database.

Output: Since the date in database is 2014-4-22T18:10:10, when the date is retrieved, OData converts and formats the time in milliseconds under UTC timezone. The retrieved value is /Date(1398190210000)/

Note

Some modules just store the date and truncate the time. For this case, for the input string 2014-4-22T18:10:10, the stored value would be 2014-4-22T00:00:00, and the retrieved value might be 2014-4-22T00:00:00

Query Example for DateTime

The property `lastModifiedWithTZ` exposes handling of date and time at an entity level.

The following API call returns the most recent handling of date and time in Java data format:

```
https://<hostname>/odata/v2/User?$format=json&$select=lastModifiedWithTZ&
$filter=lastModifiedWithTZ ge datetime'2013-12-18T17:19:28'&$top=200
```

Part of the response is:

```
"uri" : "https://<localhost>:443/odata/v2/User('eclark1')", "type" : "SFOData.User" },
"lastModifiedWithTZ" : "\/Date(1392046843000-0480)\/"
```

4.2.2 Using the DateTimeOffset Format

DateTimeOffset takes time zone into account when coverting dates.

Example of DateTimeOffset format (Server location is GMT-4:00)

Atom Format

Input: If input date is 2014-4-22T18:10:10-04:00, OData converts the input date to server time, because the input date has the same time zone with server time zone, there is no need for time conversion, and 2014-4-22T18:10:10 is stored.

If input date is 2014-4-22T23:10:10+01:00, OData convert the input date to server time, because the input date time zone is different from server time zone (input date is at GMT+1 time zone), the date is stored as 2014-4-22T18:10:10.

Output: If date in the database is 2014-4-22T18:10:10, it is returned as 2014-4-22T18:10:10-04:00.

JSON Format

Input:For input date 2014-4-22T18:10:10 in milliseconds of UTC is 1398219010000, so the input date string in json is "/Date(1398219010000-0240)/"

Output: If the date in the database is 2014-4-22T18:10:10, when retrieved, it is converted to milliseconds of UTC, / Date(1398219010000-0240)/.

Query Example for DateTimeOffset

The property `lastModifiedWithTZ` exposes handling of date and time at an entity level.

The following API call returns the most recent handling of date and time in Java data format:

```
https://<hostname</odata/v2/User?$format=json&$select=lastModifiedWithTZ&
$filter=lastModifiedWithTZ%20ge%20datetimeoffset%272013-12-18T17:19:28-07:00%27&
$top=200
```

Part of the response is:

```
"uri" : "https://<localhost>:443/odata/v2/User('eclark1')", "type" : "SFOData.User" },
"lastModifiedWithTZ" : "\/Date(1392046843000-0480)\/"
```

4.3 Query Operations

This topic explains the main types of query operations for the SuccessFactors HCM Suite OData API: system queries, custome queries, and pagination queries.

4.3.1 System Query

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

4.3.1.1 The $orderby Keyword

This topic shows examples of the `$orderby` option used in a URI.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

EXAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User?$orderby=username` or`http://<hostname>/odata/v2/User?$orderby=username asc`	All `User` Entries are returned in ascending order when sorted by the `username` Property.
`http://<hostname>/odata/v2/User?$orderby=username,hr/username desc`	Same as the URI above, except the set of User Entities is subsequently sorted in descending order by the username property of the related `hr` entry.

Note

$orderby and child entities

$orderby orders the top level entity only; child entities are not ordered. When using $orderby on a 1: many navigation properties, the order is not predicable.

4.3.1.2 The $top Keyword

This topic shows examples of the `$top` keyword used in a URI.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

EXAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User?$top=5`	The first 5 `User` Entries are returned where the Collection of User Entries are sorted using a schema determined by the OData service.
`http://<hostname>/odata/v2/User?$top=5&$orderby=username desc`	The first 5 User Entries are returned in descending order and sorted by the `username` property.

4.3.1.3 The $skip Keyword

This topic shows examples of the `$skip` keyword used in a URI.

TTo learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

EXAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User(1)/proxy?$skip=2`	The set of `proxy` Entries (associated with the `User` Entry identified by key value 1) starting with the third User.
`http://<hostname>/odata/v2/User?$skip=2&$top=2&$orderby=username`	The third and fourth `User` Entries from the collection of all `User` entities when the collection is sorted by `username` in ascending order.

4.3.1.4 The $filter Keyword

This topic shows examples of the logical operators, arithmetic operators, grouping operators, and customized operators including customized string functions that can be used with the `$filter` option.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

Logical Operators

LOGICAL OPERATOR	DESCRIPTION	EXAMPLE
Eq	Equal	Finds a `User` Entity whose `hr` / `username` is cgrant (`hr` is the navigation property): `/User?$filter=hr/username eq 'cgrant'`t.Finds a `User` Entity whose `username` property is cgrant: `/User?$filter=username eq 'cgrant’ .`
Ne	Not equal	`/ User?$filter=hr/username ne 'London'`
Gt	Greater than	Finds a `PickListLabel`Entity whose id is greater than 2000:`/ PicklistLabel?$filter=id gt 20000`
Ge	Greather than or equal	`/ PicklistLabel?$filter=id ge 10`
Lt	Less than	`/ PicklistLabel?$filter=id lt 20`
Le	Less than or equal	`/ PicklistLabel?$filter=id le 100`
And	Logical and	`/ PicklistLabel?$filter=id le 1005 and id gt 1000`to find PicklistLabel whose id is within range [1000,1005]
Or	Logical or	`/ PicklistLabel?$filter=id le 3.5 or id gt 200`
Not	Logical negation	`/User?$filter=not endswith(username,'grant')`

Arithmetic operators

ARITHMETIC OPERATOR	DESCRIPTION	EXAMPLE
Add	Addition	`/PicklistLabel?$filter=id add 5 gt 10`
Sub	Subtraction	`/ PicklistLabel?$filter=id sub 5 gt 10`
Mul	Multiplication	`/ PicklistLabel?$filter=id mul 2 gt 2000`
Div	Division	`/ PicklistLabel?$filter=id div 2 gt 4`

ARITHMETIC OPERATOR	DESCRIPTION	EXAMPLE
Mod	Modulo	`/ PicklistLabel?$filter=id mod 2 eq 0`

Grouping operator

GROUPING OPERATOR	DESCRIPTION	EXAMPLE
()	Precedence grouping.	`/ PicklistLabel?$filter=(id sub 5) gt 10`

Customized operators

CUSTOMIZED OPERA-TOR	DESCRIPTION	EXAMPLE
In	In clause	Identifies `User` Entities whose `userId` is equal to one specified by an In clause: `/User?$filter=userId in 'ctse1','mhuang1','flynch1'&$select=username,userId`

CUSTOMIZED OPERA-TOR	DESCRIPTION	EXAMPLE
like	Like clause	Identifies `User` Entities whose `userId` property “cgrant” may be retrieved with a Like clause:`/User?$filter=userId like ’cgrant’` Identifies `User` Entitieswhose `userId` property is equivalent to “cgrant”: `$filter=userId eq 'cgrant’.` Identifies `User` entities whose `userId` property “cgrant” may be retrieved with a Like clause:`/User?$filter=userId like ’cgrant%’` Identifies `User` Entities whose `userId` property starts with "cgrant":`$filter=startswith(userId, 'cgrant’).` Identifies `User` entities whose `userId` property ends with "cgrant".`/User?$filter=userId like ’%cgrant’`, equivalent to `$filter=endswith(userId, 'cgrant’)` Identifies `User` entities whose `userId` property contains string "cgrant".`/User?$filter=userId like ’%cgrant%’` Identifies `User` entities whose `userId` property contains string "cgrant" and is case insensitive.`/User?$filter=tolower(userId) like ’%cgrant%’` Same as previous.`/User?$filter=toupper(userId) like ’%cgrant%’` Identifies `User` entities whose `userId` is like "cgrant" and unions with `username` like "cgrant".dentifies `User` entities whose `userId` is like "cgrant" and unions with `username` like "cgrant".`/User?$filter=toupper(userId) like ’%cgrant%’ or tolower(username) like '%cgrant%’` Identifies `User` entities whose `userId` is like "cgrant" and intersects with `username` like "cgrant".`User?$filter=toupper(userId) like ’%cgrant%’ and tolower(username) like '%cgrant%’`

String functions

In addition to operators, a set of functions are also defined for use with the filter query string operator. The following table lists the available functions.

Note

1. ISNULL or COALESCE operators are not defined. Instead, there is a null literal which can be used in comparisons.
2. If your query contains a single quote, it has to be escaped by another one if it appears with single-quoted string.
3. Due to the Oracle maximum number of expressions (1000) in a list (ORA-01795), OData API does not accept an IN clause exceeding more than 1000 expressions in the $filter.

STRING FUNCTION	EXAMPLE
`bool endswith(string p0, string p1)`	`OData_root/ User? $filter=endswith(username, 'Futterkiste')`
`bool startswith(string p0, string p1)`	`OData_root/ User? $filter=startswith(username, 'Alfr')`
`string tolower(string p0)`	`OData_root/ User?$filter=tolower(username) eq 'alfreds futterkiste'`
`string toupper(string p0)`	`OData_root/ User?$filter=toupper(username) eq 'ALFREDS FUTTERKISTE'`
`string trim(string p0)`	`OData_root/ User?$filter=trim(username) eq 'Alfreds Futterkiste'`

The $filter option can also be used in a query using a `datetime` parameter:

QUERY EXAMPLE	
`Odata_root/User?$format=json&$select=lastModified&$filter=lastModified%20ge%20datetime%272003-12-18T17:19:28%27&$top=200`	The query returns last modified date in Java data format (the milliseconds since January 1, 1970, 00:00:00 GMT).

4.3.1.5 The $expand Keyword

This topic shows examples of the `$expand` option used in a URI.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

EXAMPLE	DESCRIPTION
`http://<hostname>/odata/v2/User?$expand=proxy`	• Identifies the Collection of `User` entities as well as each of the `proxy` associated with each `User`. • Described by the Entity Set named "User" and the "proxy" Navigation Property on the `User` Entity Type in the service metadata document.

EXAMPLE	DESCRIPTION
`http://<hostname>/odata/v2/User?$expand=hr/matrixManager`	• Identifies the Collection of `User` entities as well as each of the `hr` associated with each `User`. In addition, the URI also identifies the `matrixManager` navigation property associated with each `hr`. • Described by the Entity Set named "User", the "hr" Navigation Property on the "User" Entity Type, and the `matrixManager` navigation Property on the "hr"entity type in the service metadata document.
`http://<hostname>/odata/v2/User?$expand=hr,proxy`	• Identifies the Collection of `User` entities as well as the `hr` and `proxy` associated with each `User`. • Described by the Entity Set named "User" as well as the "hr" and "proxy" navigation properties on the "User" Entity Type in the service metadata document.

4.3.1.6 The $format Keyword

This topic shows examples of the `$format` option used in a URI and the Response received for each one.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

$FORMAT VALUE	RESPONSE MEDIA TYPE
atom	application/atom+xml
json	application/json

The following table shows examples of the `$format` keyword:

$FORMAT	EXAMPLE RESPONSE
`http://<hostname>/odata/v2/User?$format=atom`	

```
<?xml version='1.0' encoding='utf-8'?>
<feed xmlns="http://www.w3.org/2005/
Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/
dataservices/metadata" xmlns:d="http://
schemas.microsoft.com/ado/2007/08/
dataservices" xml:base="http://
<hostname>/odata/v2/">
    <title type="text">Picklist</title>
    <id>http://<hostname>/odata/v2/
Picklist</id>
    <updated>2013-07-29T07:10:44Z</
updated>
    <link rel="self" title="Picklist"
href="Picklist" />
    <entry>
        <id>http://<hostname>/odata/v2/
Picklist('CandidateStatus')</id>
        <title type="text" />
        <updated>2013-07-29T07:10:44Z</
updated>
        <author>
            <name />
        </author>
        <link rel="edit"
title="Picklist"
href="Picklist('CandidateStatus')" />
        <link rel="http://
schemas.microsoft.com/ado/2007/08/
dataservices/related/picklistOptions"
type="application/atom+xml;type=entry"
title="picklistOptions"
href="Picklist('CandidateStatus')/
picklistOptions" />
        <category
term="SFOData.Picklist" scheme="http://
schemas.microsoft.com/ado/2007/08/
dataservices/scheme" />
        <content type="application/xml">
            <m:properties>

<d:picklistId>CandidateStatus</
d:picklistId>
            </m:properties>
        </content>
    </entry>
</feed>
```

$FORMAT	EXAMPLE RESPONSE

`http://<hostname>/odata/v2/User?$format=json`	
	```
	d": {
	        "results": [{
	            "__metadata": {
	                "uri": "http://
	<hostname>/odata/v2/
	Picklist('CandidateStatus')",
	                "type":
	"SFOData.Picklist"
	            },
	            "picklistId":
	"CandidateStatus",
	            "picklistOptions": {
	                "__deferred": {
	                    "uri":
	"Picklist('CandidateStatus')/
	picklistOptions"
	                }
	            }
	        }]
	    }
	}
	```

4.3.1.7 The $select Keyword

This topic shows examples of the `$select` option used in a URI.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

The following set of examples uses the data sample data model available at `http://<hostname>/odata/v2/$metadata` to describe the semantics for a base set of URIs using the $select system query option. From these base cases, the semantics of longer URIs are defined by composing the rules below:

EXAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User?$select=username,userId`	• `username` and `userId` property values are returned for each `User` Entry. • If the $select query option had listed a property that identified a `Complex Type`, then all properties defined on the `Complex Type` must be returned.
`http://<hostname>/odata/v2/User?$select=username,proxy`	`username` property value and a link to the related proxy Entry is returned for each User.
`http://<hostname>/odata/v2/User?$select=username,proxy&$expand=proxy/hr`	All the properties of the Entries identified by the `proxy` and `hr` Navigation Properties are returned.
`http://<hostname>/odata/v2/User?$select=username,proxy/username&$expand=proxy`	In a response from an OData service, the `username` property is included and `proxy` Entries with a `username` property is included. But, instead of including the fully expanded proxy Entries, each `proxy` contains a `user` property.

4.3.2 Custom Query

This topic shows an example of a custom query used in a URI.

To learn about the OData System query options that will be used in the example URIs in this section, visit www.odata.org, and search for "OData Version 2.0 Uri Conventions".

EXAMPLE URI	DESCRIPTION
`http://<hostname>/odata/v2/User?purge=true`	Identifies all `User` Entities. Includes a custom query Entitiy named `purge` whose meaning is service specific.

4.3.3 Examples of Queries with Multiple Keywords

Examples of queries with multiple keywords.

4.3.3.1 Query with nested-navigation properties with $select

This topic explains the use of $select in a query.

The following URIs are constructed differently but provide the same response:

`http://<hostname>/odata/v2/User?$select=userId,username,hr&$format=json&$top=1`

`http://<hostname>/odata/v2/User?$select=userId,username,hr/manager/proxy&$format=json&$top=1`

Each URI identifies all `User` Entities. Instead of having the response contain all properties for each Entity, the $select option ensures the response contains only the three properties specified in the URI. . You can specify a simple property, a navigation property, or a nested-navigation property. For a simple property, the response contains results. For a navigation property, the response is a deferred link , whatever the depth of the property.

Sample response:

```
{ "__metadata" : { "uri" : "http://localhost:8080/odata/v2/User('mhuang1')", "type" :
"SFOData.User" }, "userId" : "mhuang1", "username" : "mhuang", "manager" :
{ "__deferred" : { "uri" : "http://localhost:8080/odata/v2/User('mhuang1')/
manager" } } }
```

4.3.3.2 Query with $select and $expand

The following example shows the `$select` and `$expand` options used together in a URI:

```
http://<hostname>/odata/v2/User?$select=userId,manager/hr/manager&$format=json&
$expand=manager/hr
```

The URI identifies all entities of a `User` Entity Set. The response is similar to the response from a URI that uses nested-navigation properties with the `$select` option, but the `$expand` option can make a difference in the results that appear in the response. For example:

- If a navigation property gets expanded, the property shows as a flat object.
- If a navigation property is not expanded, it shows as a deferred link.
- If the nested-navigation property `manager/hr/manager` gets expanded, it shows a manager object.
- Because the last entry in a nested navigation doesn't get expanded, the manager object is given a nested `hr` property. The `hr` property will have a deferred link named *manager* which refers to the actual object. Unselected properties of expanded objects will be trimmed..

Sample Response:

```
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_200')", "type" :
"SFOData.User"
}, "userId" : "asliva_upsert_JAM_200", "manager" : {
    "__metadata" : {
    "uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_201')",
"type" : "SFOData.User"
    },
    "hr" : {
            "__metadata" : {
            "uri" : "http://localhost:8080/odata/v2/
User('asliva_upsert_b11_112_inline_manager11')", "type" : "SFOData.User"
            },
            "manager" : {
                "__deferred" : {
                    "uri" : "http://localhost:8080/odata/v2/
User('asliva_upsert_b11_112_inline_manager11')/manager"
                    }
                }
        }
    }
```

4.3.3.3 Query links with $select and $expand

The following example shows $select with $expand:

```
http://<hostname>/odata/v2/User('asliva_upsert_JAM_200')/proxy?
$select=userId,manager/hr/manager&$format=json&$expand=manager/hr
```

In the response, the `$expand` option expands the selected navigation property. Unselected data columns do not get expanded. Set entities of `User/proxy` are returned.

Sample Response:

```
[
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_197')", "type" :
"SFOData.User"
}, "userId" : "asliva_upsert_JAM_197", "manager" : null
},
{
"__metadata" : {
```

```
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_198')", "type" :
"SFOData.User"
}, "userId" : "asliva_upsert_JAM_198", "manager" : null
},
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/
User('asliva_upsert_b11_112_inline_proxy')", "type" : "SFOData.User"
}, "userId" : "asliva_upsert_b11_112_inline_proxy", "manager" : null
},
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/
User('asliva_upsert_b11_112_inline_proxy_1')", "type" : "SFOData.User"
}, "userId" : "asliva_upsert_b11_112_inline_proxy_1", "manager" : null
}
```

4.3.3.4 Query with $select and $expand and $filter

The following example shows a query with multiple options:

```
 http://<hostname>/odata/v2/User?$select=userId,manager/hr/manager&$format=json&
$expand=manager/hr&$filter=proxy/userId eq 'asliva_upsert_b11_112_inline_proxy_1'
```

The response identifies a set of `User` Entities whose `proxy` property is `userId` , which is equal to `asliva_upsert_b11_112_inline_proxy_1`. After the entities are retrieved, the response removes unselected columns. Then the `$expand` option expands the selected columns that is a navigation property..

Sample Response:

```
[
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_200')", "type" :
"SFOData.User"
}, "userId" : "asliva_upsert_JAM_200", "manager" : {
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_201')", "type" :
"SFOData.User"
}, , "hr" : {
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/
User('asliva_upsert_b11_112_inline_manager11')", "type" : "SFOData.User"
}, , "manager" : {
"__deferred" : {
"uri" : "http://localhost:8080/odata/v2/
User('asliva_upsert_b11_112_inline_manager11')/manager"
}
}
}
}
},
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_201')", "type" :
"SFOData.User"
}, "userId" : "asliva_upsert_JAM_201", "manager" : {
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('asliva_upsert_JAM_198')", "type" :
"SFOData.User"
}, "hr" : null
}
}
```

4.3.3.5 Query with $top and $skip

Using the `$top` and `$skip` options together in a request URI creates a pagination query. The response is a subset of the whole result, from `$skip` to `$top`. `$skip` indicates the starting row for the query. `$top` limits the size of the query.

For example, if the URI uses `$top=50, $skip=20`, the response is a subset from numbers 21 to number 70 of the whole result set.

4.3.4 Pagination

The OData API provides simple pagination of query results.

Pagination limits the maximum result size of a query response to 1000 records. The OData standard provides a `'__next'` link in your query response if there are more results in the database.

Best practices: For best performance when paginating across many records, the page size ($top) should be as close to the 1000 limit as possible. But the time to execute a single GET should complete within the http client timeout time or else an http timeout error could occur. Most clients allow times of 2 to 5 minutes. Therefore the $top value should be as large as possible but not so large that there is a risk of the request responding with a non-200 timeout error.

The following example shows a URI whose response uses pagination:

`GET http://<hostname>/odata/v2/User?$format=json`.

Sample response:

```
{ "d" : { "results" : ................ "__next" : "http:// <hostname>/odata/v2/User?
$format=json&$skiptoken=eyJzdGFydFJvdyI6NSwiZW5kUm93IjoxMH0%3D" } }
```

The `$skiptoken` option indicates the `startRow` and `endRow` of the Entity records for your next query. Copy the URL and access it to get the next 1000 records.

4.3.5 Querying Effective-Dated Entities

You can use the following additonal keywords for effective-dated entities (such as most Employee central entities and those MDF entities that are created effective-dated) :

Keyword	API Call
asOfDate	`https://<hostname>/odata/v1/EmpJob&$format=json&asOfDate=2014-01-01`
fromDate/toDate	`https://<hostname>/odata/v1/EmpJob&$format=json&fromDate=2014-01-01&toDate=2014-12-31`

How effective-dating is handled in an OData API query

The following rules are followed when an effective dated entity is queried:

1. If both the base entity and the navigation entity are effective-dated, then when expanding the navigation entity, the effectiveStartDate of the base entity is used as the asOfDate of the navigation entity.
2. If the base entity is none effective-dated, TODAY is used as asOfDate when expanding the navigation entity.
3. For a Picklist field, the navigation entity is PicklistValue. When expanding a Picklist field, the effectiveStartDate/effectiveEndDate of the Picklist entity are used because PicklistValue itself is not effective-dated.
4. If you specify asOfDate as a query option, all entities in all levels of the query use this date as asOfDate.
5. If you specify fromDate and toDate as query options, then only the top level entity is filtered by the fromDate and toDate, all lower level entities follow rules 1, 2, and 3.
6. You can use either AsOfDate or fromDate/endDate in a query request, not both.

4.4 Create Operation

This section describes the `Create` operation.

To learn about OData Operations, visit www.odata.org, and search for "OData Version 2.0 Operations".

The following table shows a sample request and response. The request uses the `Create` operation:

SAMPLE JSON REQUEST	SAMPLE JSON RESPONSE

```
POST /odata/v2/User Host: <hostname>
accept: application/json content-type:
application/json
POST data:
{
    "__metadata": {
        "uri": "User('NewUser')"
    },
    "username": "NewUser",
    "password": "pwd",
    "hireDate": "/Date(978307200000)/",
    "gender": "M",
    "status": "active",
    "userId": "NewUser",
    "firstName": "Paul",
    "lastName": "Chris",
    "email": user@successfactors.com",
    "department": "Retail Banking",
    "timeZone": "PST",
    "hr": {
        "__metadata": {
            "uri": "User('HRUser')"
        }
    },
    "manager": {
        "__metadata": {
            "uri": "User('NewManager')"
        },
        "username": "NewManager",
        "password": "pwd",
        "hireDate": "/
Date(978307200000)/",
        "gender": "M",
        "status": "active",
        "userId": "NewManager",
        "firstName": "Paul",
        "lastName": "Chris",
        "email":
"user@successfactors.com",
        "department": "Retail Banking",
        "timeZone": "PST",
        "hr": {
            "__metadata": {
                "uri": "User('HRUser')"
            }
        },
        "manager": {
            "__metadata": {
                "uri":
"User('OldManager')"
            }
        }
    }
}
```
	```
	HTTP/1.1 201 Created
	Location: http://<hostname>/odata/v2/
	User('NewUser')
	DataServiceVersion: 1.0
	Content-Type: application/json;
	charset=utf-8
	```

4.5 Update Operation

The `Update` operation can either replace existing data or merge new data with the already existing data.

4.5.1 Replacing Existing Data

To update the data in an Entity, clients execute an HTTP `PUT` request. The new data is included in the request body. The `PUT` operation replaces the existing data in an entity, so all property values in the Entity either take the values provided in the request body or are reset to their default value if no data is provided in the request.

The `PUT` request returns status `204` (No Content) to indicate success. oNo response body is returned. The following table shows a sample request and response:

SAMPLE JSON REQUEST	SAMPLE JSON RESPONSE
`PUT /odata/v2/User('NewUser’) Host: <hostname> accept: application/json content-type: application/json PUT data: { "__metadata": { "uri": "User('ExistUser')" }, "username": "ExistUser", "password": "pwd", "hireDate": "/Date(978307200000)/", "gender": "M", "status": "active", "userId": "ExistUser", "firstName": "Paul", "lastName": "Chris", "email": "user@successfactors.com", "department": "Retail Banking", "timeZone": "PST", "hr": { "__metadata": { "uri": "User('HRUser')" } }, "manager": { "__metadata": { "uri": "User('OldManager')" } } }`	`HTTP/1.1 200 OK DataServiceVersion: 1.0`

```
PUT /odata/v2/User('NewUser’) Host:
<hostname> accept: application/json
content-type: application/json
PUT data:
{
    "__metadata": {
        "uri": "User('ExistUser')"
    },
    "username": "ExistUser",
    "password": "pwd",
    "hireDate": "/Date(978307200000)/",
    "gender": "M",
    "status": "active",
    "userId": "ExistUser",
    "firstName": "Paul",
    "lastName": "Chris",
    "email": "user@successfactors.com",
    "department": "Retail Banking",
    "timeZone": "PST",
    "hr": {
        "__metadata": {
            "uri": "User('HRUser')"
        }
    },
    "manager": {
        "__metadata": {
            "uri": "User('OldManager')"
        }
    }
}
```

```
HTTP/1.1 200 OK
DataServiceVersion: 1.0
```

4.5.2 Merging with Existing Data

In certain cases, you might want to do an incremental update without replacing all the content of a data entity. To avoid overloading the`PUT` operation, OData offers a custom HTTP `MERGE`. A `MERGE` request updates only the properties provided in the request body, and leaves the data not mentioned in the request body in its current state.

The MERGE request is delivered using the POST HTTP operation with a header indicating the method is MERGE, as shown in the table beneath.The `MERGE` request returns status `200 OK` (No Content) to indicate success. No, response body is returned. The following table shows a sample request and response:

SAMPLE JSON REQUEST: MERGE	SAMPLE JSON RESPONSE (Status 200)

`POST /odata/v2/`
`User('ExistingUser')content-type:`
`application/json: x-http-method: POST`
`{`
`    "email":`
`"newMail@successfactors.com"`
`}`
	`HTTP/1.1 200 OK`
	`DataServiceVersion: 1.0`

For merging data that includes navigation properties, take a look at the examples in Merging Existing Links [page 44]

4.6 Upsert Operation

The `Upsert` operation is an OData Service Operation. It takes an array of Entities as the request. The server updates the Entity for which an external id already exists. A new Entity is inserted if the external id doesn't exist. Inline properties of Entities are also updated. Clients can invoke the `Upsert` operation using the HTTP `POST` request.

The operation always returns status `200 (OK)` with a response body indicating all `Upsert` results. Individual `Upsert` failures are not reported as errors. The following table shows a sample request and response:

SAMPLE JSON REQUEST

```
POST /odata/v2/upsert Host:<hostname>
accept: application/json content-type:
application/json;charset=utf-8
POST data:
[
    {
        "__metadata": {
            "uri": "User('NewUser')"
        },
        "username": "NewUser",
        "password": "pwd",
        "hireDate": "/
Date(978307200000)/",
        "gender": "M",
        "status": "active",
        "userId": "NewUser",
        "firstName": "Paul",
        "lastName": "Chris",
        "email":
"user@successfactors.com",
        "department": "Retail Banking",
        "timeZone": "PST",
        "hr": {
            "__metadata": {
                "uri": "User('HRUser')"
            }
        },
        "manager": {
            "__metadata": {
                "uri":
"User('OldManager')"
            }
        }
    },
    {
        "__metadata": {
            "uri": "User('ExistUser')"
        },
        "username": "ExistUser",
        "password": "pwd",
        "hireDate": "/
Date(978307200000)/",
        "gender": "M",
        "status": "active",
        "userId": "ExistUser",
        "firstName": "Chris",
        "lastName": "Paul",
        "email":
"user@successfactors.com",
        "department": "Retail Banking",
        "timeZone": "PST",
        "hr": {
            "__metadata": {
                "uri": "User('HRUser')"
            }
        },
        "manager": {
            "__metadata": {
                "uri":
"User('OldManager')"
            },
            "username": "OldManager",
            "password": "pwd",
```

SAMPLE JSON RESPONSE

```
HTTP/1.1 200 OK
DataServiceService: 1.0
Content-Type: application/
json;charset=utf-8
Response data:
{
    "d": [
        {
            "key": "NewUser",
            "status": "OK",
            "editStatus": "INSERTED",
            "message": null,
            "index": "0",
            "inlineResults": null
        },
        {
            "key": "ExistUser",
            "status": "OK",
            "editStatus": "UPDATED",
            "message": null,
            "index": "1",
            "inlineResults": [
                {
                    "results": [
                        {
                            "key":
"OldManager",
                            "status":
"OK",

"editStatus": "UPDATED",
                            "message":
null,
                            "index":
"0",

"inlineResults": null
                        }
                    ],
                    "inlineProperty":
"manager"
                }
            ]
        }
    ]
}
```

SAMPLE JSON REQUEST	SAMPLE JSON RESPONSE

```
            "hireDate": "/
Date(978307200000)/",
            "gender": "F",
            "status": "active",
            "userId": "OldManager",
            "firstName": "Paul",
            "lastName": "Chris",
            "email":
"user@successfactors.com",
            "department": "Retail
Banking",
            "timeZone": "PST",
            "hr": {
                "__metadata": {
                    "uri":
"User('HRUser')"
                }
            },
            "manager": {
                "__metadata": {
                    "uri":
"User('AnotherManager')"
                }
            }
        }
    }
]
```

4.6.1 Upsert parameter: strictTransactionIsolate

strictTransactionIsolate is an upsert parameter for determining if upsertSingle, upsertMultiple, or changeset rollback is applied.

UpsertSingle allows the success of records even if some records fail in a multiple record upsert. UpsertSingle is used for records with inline properties and records that do not support full purge.

UpsertMultiple is called when UpsertSingle does not apply.

Changeset rollback reverts the entire transaction when there is a partial upsert failure.

The value of *strictTransactionIsolate* can be applied in the following use cases:

- Multiple record upserts
- Upserts of entities that do not support full purge

Desired multiple record upsert behavior	Set strictTransactionIsolate =	Additional Information
Standard upsert behavior	false (or leave the value blank)	-
UpsertSingle	true	Records are independent of each other (catch java.lang.Exception)

Desired multiple record upsert behavior	Set strictTransactionIsolate =	Additional Information
Rollback changeset in $batch	true	When one record fails, changeset will rollback
Full purge entities, upsertMultiple	true	This parameter makes no sense for entities that do not support full purge

Related Information

Upsert parameter: strictTransactionIsolate [page 40]

4.6.2 $batch: Upsert and changeset behavior

In contrast to the standard OData API $batch processing specification, a partial upsert failure reverts the transaction for the current changeset.

In the standard OData API $batch calls, the changeset response code is HTTP 200 even when one or more of the records in the payload fail. In a use case such as new hire, using $batch with this standard behavior means that some users are not created, some are created but in either case HTTP 200 is the resulting changeset response code and behavior.

In the upsert request, the parameter strictTransactionIsolate lets you influence this behavior. When strictTransactionIsolate=true for a $batch upsert multiple records request, the behavior is as follows:

- One record fails in the request → all other records rollback
- First record fails in the request → First failure message in $batch changeset response
- Failed record/records in request → Changeset response code is HTTP 500
- Failed record/records → Index in changeset response identifies the failed request number in the changeset
- Failed record/records → all changes before failure rollback; following request in changeset is terminated (each changeset in the whole $batch request is atomic)
- Each changeset in a whole $batch request is independent; one changeset rollback does not affect others.

Note

The behavior of a normal upsert remains → when upserting multiple records and one fails, the other records work and HTTP 200 is the response code.

Related Information

Upsert parameter: strictTransactionIsolate [page 40]

4.7 Working with Links

4.7.1 Creating Links

You can create a new link between Entities by referencing one Entity during the creation or modification of another, or by explicitly issuing a `POST` request against the URL of the `Link` option. The request must have the new URI in the request body following the appropriate format for stand-alone `Link` options in XML or JSON. Upon success, the operation returns status `204 No Content`. No response body is returned.

To create a `Link`, these rules must be followed:

1. A new `Link` is always inserted for a 1: N relation. The following request adds a `Link` to navigate from an existing `User` Entity to another existing `User` by the `matrixManager` property:

SAMPLE REQUEST	SAMPLE RESPONSE

`POST http://<hostname>/odata/v2/User('adam_07031_101_1')/$links/matrixManager`
`POST data:`
`<uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices">`
`http://<hostname>/odata/v2/User('asliva__701_01')`
`</uri>`
	`HTTP/1.1 204 No Content`

2. The server always performs a `MERGE` operation for a 1:1 relationship. The following example shows that for a 1:1 relationship, if a target link does not exist, a new link is inserted. If the target link already exists, the existing link is removed and a new link is inserted:

SAMPLE REQUEST	SAMPLE RESPONSE

`POST http://<hostname>/odata/v2/User('adam_07031_101_1')/$links/secondManager`
`POST data:`
`<uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices">`
`http://<hostname>/odata/v2/User('asliva__701_03')`
`</uri>`
	`HTTP/1.1 204 No Content`

4.7.2 Deleting Links

You can remove existing links using a `DELETE` request against the `Link` option in a URI. Upon success, the operation returns status `204` . No response body is returned.For example, the following request deletes the `Link` option between a `User` Entity and its containing `matrixManager` property:

`DELETE /odata/v2/User('adam_07031_101_1')/$links/matrixManager('asliva__701_01') HTTP/1.1` You can specify a key to remove single link. All links are removed if no key is provided.

4.7.3 Updating Links

You can update existing `Link` options by either replacing or merging them.

4.7.3.1 Replacing Existing Links

You can replace a `Link` option with a new link using an HTTP `PUT` request against the `Link` option in a URI, with the new URI in the request body following the appropriate format for stand-alone links in XML or JSON. The operation returns status `204 No Content`.

To update a `Link`, these rules must be followed:

1. For a 1: N relationship, you must specify a key to update a specific link. The following example replaces an existing link `matrixManager`('asliva__701_01') property with `matrixManager`('asliva__701_02'). The result is that the `User` Entity('adam_07031_101_1') has a link pointing to `matrixManager`('asliva__701_02'):

SAMPLE REQUEST	SAMPLE RESPONSE

`PUT http://<hostname>/odata/v2/User('adam_07031_101_1')/$links/matrixManager('asliva__701_01')`
`PUT data:`
`<uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices">`
`http://<hostname/odata/v2/User('asliva__701_02')`
`</uri>`
	`HTTP/1.1 204 No Content`

2. For 1:1 relation, you cannot specify the key to update. When a server receives a delete request for 1:1 link, it first checks to see if the link exists. If the link exists, it removes the existing link and inserts a new link. If no link exists a new link is inserted.

SAMPLE REQUEST	SAMPLE RESPONSE
`PUT http://<hostname>/odata/v2/` `User('adam_07031_101_1')/$links/hr` `PUT data:` `<uri xmlns="http://` `schemas.microsoft.com/ado/2007/08/` `dataservices">` `http://<hostname/odata/v2/` `User('asliva__701_02')` `</uri>`	`HTTP/1.1 204 No Content`

4.7.3.2 Merging Existing Links

To merge `Links`, these rules must be followed:

1. For a 1:N multiplicity in association, a new link is added to the existing Entity. This example adds a new `MatrixManager` property:

SAMPLE REQUEST	SAMPLE RESPONSE
`POST http://<hostname>/odata/v2/` `User('adam_07031_101_1')` `POST data:` `{` `"matrixManager":` `{ "__metadata": { "uri":` `"User('asliva__701_03')" }` `}`	`HTTP/1.1 200 OK`

2. For a 1:1 relationship, the existing link is replaced by a new link:

SAMPLE REQUEST	SAMPLE RESPONSE
`POST http://<hostname>/odata/v2/` `User('adam_07031_101_1')` `POST data:` `{` `"hr":{ "__metadata":` `{ "uri": "User('asliva__701_01')" }` `}`	`HTTP/1.1 200 OK`

4.8 OData $Batch Operation

The OData protocol has introduced T operation that can combine multiple requests into a single batch request. The OData Batch Request is represented as a Multipart MIME v1.0 message, a standard format allowing the representation of multiple parts, each of which may have a different content type, within a single request.

Batch Request Header

Batch request are submitted as a single POST request to the `/odata/v2/$batch` URL. The request must include a content-type header specifying a content type of "multipart/mixed" and a boundary specification. This is an example for a batch request header:

Sample Code

```
POST /odata/v2/(restricted/internal)/$batch
HOST: localhost
Content-Type: multipart/mixed; boundary=batch_36522ad7-fc75-4b56-8c71-56071383e77b
```

The batch boundary needs to follow the pattern `[a-zA-Z0-9_\\-\\.'\\+]{1,70}` and it must not contain `( ) < > @ , ; : / " [ ] ? =` special characters.

Batch Request Body

In a batch request, each ChangeSet and query request is represented as a distinct MIME part and is separated by the boundary maker defined in the Content-Type. This is an example of a batch request body that includes:

1. Query the user "admin".
2. ChangeSet: Insert a new user.
3. Replace a user, "carla"
4. Query the Attachment

Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding:binary
GET User('admin') HTTP/1.1
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST User HTTP/1.1
Content-Type: application/atom+xml;type=entry
<AtomPub representation of a new User>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
```

```
Content-Transfer-Encoding:binary
PUT User('carla') HTTP/1.1
Content-Type: application/json
<JSON representation of User carla>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding:binary
GET Attachment(“invalidAttachmentId”) HTTP/1.1
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

Batch Processing

Request parts in a batch are processed sequentially and independantly of each other. A request part can be a Query request or a ChangeSet. All operations in a single ChangeSet unit follow the all or none rule. Failure of a single unit causes the entire ChangeSet to fail. No partial changes are applied.

Referencing Requests in a Change Set

If there is an Insert request in the ChangeSet and the MIME part representing that request includes a Content-ID header, then the new entity that is created by the insert operation can be referenced by subsequest requests within that ChangeSet by referring to the Content-ID value prefixed with a “$” character:

Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST User HTTP/1.1
Content-Type: application/atom+xml;type=entry
Content-ID: 1
<AtomPub representation of a new User>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST $1 /Manager HTTP/1.1
Content-Type: application/atom+xml;type=entry
<AtomPub representation of a new user manager>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

This batch request:

1. Inserts a new User
2. Insert the User’s Manager

Batch Response Header

The body of a batch response contains a response for each Query and ChangeSet request that makes up the batch request. The order of reponses matches the order of requests. Each response includes a Content-Type header with a value of "application/http", and a and a Content-Transfer-Encoding MIME header with a value of “binary”.

Sample Code

```
-batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding: binary
HTTP/1.1 200 Ok
Content-Type: application/atom+xml;type=entry
Content-Length: ###
<AtomPub representation of the User entity with EntityKey admin>
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
Content-Length: ###
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
HTTP/1.1 201 Created
Content-Type: application/atom+xml;type=entry
Location: http://localhost/odata/V2/User('abc')
Content-Length: ###
<AtomPub representation of a new User entity>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
HTTP/1.1 204 No Content
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding: binary
HTTP/1.1 404 Not Found
Content-Type: application/xml
Content-Length: ###

<Error message>
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

Note

For a successful ChangeSet request, the response lists all the request responses included in the ChangeSet. For example, if a ChangeSet has three requests, then the response contains three responses in the ChangeSet body. However, if the changeset fails, then only one response entry, which is the error in the problematic request is displayed. The ChangeSet response is then a non-ChangeSet response because it does not have the ChangeSet body.

Related Information

http://www.odata.org/documentation/odata-version-3-0/batch-processing/

4.8.1 Examples of $batch Request Bodies

Examples of $batch requests are listed. Please strictly follow the format in the examples including the new lines and spaces.

Example of a Single Query

Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding:binary
GET http://localhost:8080/odata/v2/User?$select=username&$top=2 HTTP/1.1
Content-Type: application/json;type=entry

--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

Example of a Single Edit

A change (insert,replace,merge,delete, upsert) request must be wrapped in a changeset.

Sample Code

```
-batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
Content-Length: 100
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST /User('admin') HTTP/1.1
Host: host
Content-Type: application/json;charset=utf-8
X-HTTP-METHOD: MERGE
{
"nickname":"newname"
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

Example of an Edit + Upsert

An upsert request is a special kind of the update request. It should be included in a ChangeSet.

Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Length: 100
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json;charset=utf-8
accept:application/json
{
    "__metadata": {
        "uri": "User('content1')"
    },
    "username": "content1",
    "password": "content1",
    "hireDate": "/Date(978307200000)/",
    "status": "active",
    "userId": "content1",
    "firstName": "content1",
    "lastName": "helloterry@abc.com",
    "department": "test",
    "timeZone": "PST",
    "hr": {
        "__metadata": {
            "uri": "User('admin')"
        }
    },
    "manager": {
        "__metadata": {
            "uri": "User('admin')"
        }
    }
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json;charset=utf-8
accept:application/json
{
    "__metadata": {
        "uri": "User('content3')"
    },
    "lastModified": "/Date(3923749827424)/"
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--

--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

Example of a Query + Edit + Upsert

Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding:binary
Content-ID: 1
```

```
GET User('admin')?$format=json HTTP/1.1
Content-Type: application/atom+xml;type=entry

--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Length: 100
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json;charset=utf-8
accept:application/json
{
    "__metadata": {
        "uri": "User('content1')"
    },
    "username": "content1",
    "password": "content1",
    "hireDate": "/Date(978307200000)/",
    "status": "active",
    "userId": "content1",
    "firstName": "content1",
    "lastName": "helloterry@abc.com",
    "department": "test",
    "timeZone": "PST",
    "hr": {
        "__metadata": {
            "uri": "User('admin')"
        }
    },
    "manager": {
        "__metadata": {
            "uri": "User('admin')"
        }
    }
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST User('content1') HTTP/1.1
Content-Type: application/json;charset=utf-8
X-HTTP-METHOD: MERGE
{
    "manager": {
        "__metadata": {
            "uri": "User('admin')"
        }
    }
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--

--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

5 Access Permissions for Entities

Two types of permisions can be set at the entity level:

- Permission to Query.
- Permission to Edit.

5.1 Permission to Query

If you are not in RBP mode, you can query all Entities as long as the *Export* permission is enabled. For RBP-based systems, there are three query permission levels:

PERMISSION LEVEL	DESCRIPTION
Operation Level	An Operation-level permission check determines if a user has the right to access the module to which the entities belong.
Row Level	A Row-level permission check determines if the logged-in user has permission to query the record in the query result set.
Field Level	A Field-level permission check determines if the logged-in user has permission to query a specific property in the query result.

:The *Export* permission follows these rules:

PERMISSION SETTINGS	QUERY OUTCOME
Export permission is enabled and query entity permission has no target population.	No row-level permission check is required. The user can query all entities.
Export permission is enabled and query entity permission has a target population.	Row-level permission check is required and only entities in the target population are in the query result.
Export permission is disabled.	Field-level permission check is performed and only fields that the user has permission to access are returned in the query result.

5.2 Permission to Edit

If you are not in RBP mode, you can edit all Entities as long as the *Import* permission is enabled. For RBP-based systems, there are three query permission levels:

PERMISSION LEVEL	DESCRIPTION
Operation Level	An Operation-level determines if user has the right to access the module to which the entities belong.
Row Level	A Row-level permission check determines if the logged-in user has permission to edit the record in the query result set.
Field Level	A Field-level permission check determines if the logged-in user has permission to edit a specific property in the query result.

: The *Import* permission follows these rules:

PERMISSION SETTINGS	QUERY OUTCOME
Import permission is enabled.	The user can edit all entities.
Import permission is disabled and the operation type is `Insert/Upsert`.	The operation cannot be executed.
Import permission is disabled and the operation type is `Update/Merge`.	A Field-level permission check is performed and only fields that the user has permission to edit can be edited. Note that Entities that have a target population also require a row-level permission check before field-level permissions are checked.

6 Integration Tools

The following integration tools are currently available for SuccessFactors HCM Suite OData APIs.

OData API Audit Log

The OData Audit Log can help with support and debugging issues related to OData usage . The Audit Log Integration tool monitors traffic, gets detailed payload information about API requests made to your system. The OData Audit log captures payload details for the last 10,000 OData calls. This log lets you inspect the exact OData payload requests made to the system and corresponding OData responses sent by the system.

The tool is for developers who use OData during an implementation, or administrators who want to share information in this log with SuccessFactors HCM Suite Support to help resolve OData related issues. You can download data from individual calls, and send it to a Support representative for analysis.

> i Note
>
> The OData Audit Log allows access to potentially sensitive data. Limit access to only trusted users.

OData API Option Profile

The OData API Option Profile lets you specify processing parameters for an API using configuration settings. Configuration settings can help reduce complexity of an integration for a client by presetting the business logic on the HCM side . This tool is currently available for the `User` Entity.

6.1 Granting Permissions for Integration Tools

Permission to access any of the *Integration Tools* pages must be granted for each individual tool by an administrator of your SuccessFactors HCM Suite system. Granting access to the *Integration Tools* pages depends on which of the following permission systems you are using in your SuccessFactors HCM Suite instance:

- User-based permission system
- Role-based permission system

6.1.1 Granting User-Based Permission to Integration Tools

To enable users to access the Integration Tools on a user-based permission system, users must be granted permission by a SuccessFactors HCM Suite administrator.

Prerequisites

To grant permissions to users, you must have administrator rights on your SuccessFactors HCM Suite system.

Context

Permission to access the Integration Tools can be found in the *Administrative Privileges* page. The location of this page depends on which version of the Administration Tools you are using.

Procedure

1. Access the *Administrative Privileges* page:
    - If you are using the new Administration Tools, select *Manage Employees* > *Set User Permissions* > *Administrative Privileges*.
    - If you are using the old Administration Tools, select *Manage Security* > *Administrative Privileges*.
2. Search for the user to grant access to, and select the checkbox next to the user's name.
3. Select the Integration Tools link.
4. To grant access the desired Integration Tool, select the appropriate checkbox .
5. Select *Save Admin Permission for Selected Users*.

6.1.2 Granting Role Based Permission to Integration Tools

Permission to access the Integration Tools is granted by an administrator of your SuccessFactors instance.

You can access the Integration Tools page from *Permission Settings* > *Manage Integration Tools*.

On the page, check the Integration Tools for which you wish to provide access.

6.2 Accessing Integration Tools

When you have permissions to access the Integration tools, you can access them as follows:

- In the old admin tool, you can access the tools from the *Integration Tools*section.
- In the new admin tool, you can access the tools from the *Company Processes & Cycles* > *Company Settings* link to view available integration tools.

6.3 Managing OData API Option Profile

You can manage account settings for a User Entity by accessing from the users's stored profile .

To access a user's profile, go to Admin Center and search for *Manage API Option Profile* page. From this page you can manage profiles. For profiles, you can enter information such as the user name, automatic process owner change and so on.

Using a Saved API Option Profile

If you use a saved API profile, add the parameter, `apiOptionProfileID` at the end of an insert/upsert URL. Currently this functionality is available only for the User entity as shown:

`http://<hostname>/odata/v2/User?apiOptionProfileID=option001` where `option001` is a saved option profile. Following is the list of currently supported parameters:

PARAMETER	DESCRIPTION
`sendWelcomeMessage`	Send welcome email to new users.
`validateMgrHr`	Validate `Manager` and `HR` fields.
`managerTransfer`	Automatically insert new manageras next document recipient if not already.
`routeInboxDoc`	Manager – Automatically transfer inbox documents to new manager.
`routeEnRouteDoc`	Manager – Automatically route enroute documents to new manager.
`routeCompletedDoc`	Manager –Automatically route copy of completed documents to new manager.
`managerOnlyCompany`	Enforce a Manager-only implementation.

PARAMETER	DESCRIPTION
`removeInProgressDocsForInactiveUsers`	Remove documents with status In-Progress for Inactive Employees.
`removeCompletedDocsForInactiveUsers`	Remove documents with status Completed for Inactive Employees .
`defaultPasswordField`	Set a new user default password field.

6.4 API Center

Describes the features of the API Center and required permissions .

The API center is a one-stop shop for accessing OData API tools. OData API tools include tools such as the OData API Audit Log, SFAPI Audit Log, OData API Version Control, and the OData API Dictionary.

You'll find the API Center in the Admin Center. If you can't see it there, check that you have the permission for at least one of the tools hosted on the API Center. From the Admin Center, choose *Set User Permissions* > *Manage Permission Roles*, and choose the Permission Role. Then choose *Permission* > *Administrator Permission* > *Manage Integration Tools*. Choose the permission required, for example *Access to OData API Data Dictionary* or *Access to OData API Version Control*.

The API Center currently hosts the OData API Audit Log, OData API Dictionary, OData IP Whitelisting, OData API Audit Log, Legacy SFAPI Audit Log, Legacy SFAPI Data Dictionary, Legacy SFAPI Metering Details, Legacy SFAPI IP Whitelisting, OAuthConfiguration for OData, OData API Version Control, and Manage User API Options.

Note

Users can only access tools for which they have permission. Otherwise the tools are displayed but they are greyed out.

Related Information

OData API Data Dictionary [page 56]

6.4.1 OData API Data Dictionary

Describes the features, required permissions, and performance tips for the OData API Data Dictionary

In the OData API data dictionary, you can see information about attributes and supported operations.

The look and feel of the OData API data dictionary has been enhanced. It's available in the Admin Center. If you can't see it there, check that you have the permission, *Access to OData API Dictionary* (available in *Manage Integration Tools* and applicable to both RBP and non-RBP users).

Note

To optimize performance, you can request that Customer Services activate the provisioning switch, *Enable OData API Dictionary Cache* (in *Web Services*).

In the OData API data dictionary, you can use *Search All*, or narrow your search to *Entity*, *Complex Type*, or *Function Import*.

Using the search filters *Entity*, *Complex Type*, or *Function Import* also lets you narrow your search by *Tag*, for example EC - Payment Information . This then ensures that under *Name*, you only see the entities available for this tag.

Information is displayed in a tabular form. A horizontal and vertical scroll bar are available to make viewing easier. The following columns are available and can be sorted alphabetically in ascending or descending order:

- *Property Name*
  In addition to sorting alphabetically, you can also filter by a defined term.
- *Property Label*
- *Type*

These columns are fixed. However, the following columns can be arranged to meet your requirements:

- *sap.picklist*
- *Business Key*
- *Nullable*
- *sap: required*
- *sap: creatable*
- *sap: updatable*
- *sap: upsertable*
- *sap:visible*
- *sap: sortable*
- *sap: filterable*
- *sap:semantics*
- *sap:display-format*
- *sap:aggregation role*
- *DefaultValue*
- *MaxLength*
- *Precision*
- *Scale*
- *sap:inlineRequired*
- *Navigation Target*

6.4.2 OData API Audit Log

Describes the features, and required permissions for the OData API Audit Log.

In the OData API Audit Log, you can display audit log information such as the time a request was made, the time taken for the response, the type of OData API call, and so on.

The look and feel of the OData API audit log has been enhanced. It's available in the Admin Center. If you can't see it there, check that you have the permission, *Access to OData API Audit Log* (available in *Manage Integration Tools* and applicable to both RBP and non-RBP users).

Because the Audit Log may contain sensitive data, ensure that access to the Audit Log is given only to trusted users. One safety strategy would be to grant permission to developers only when necessary for development and debugging purposes, and then remove access to this feature.

Note

Data is captured in the Audit Log whether any user has permission to view the log or not. Therefore, removing a user's access to the log will not remove the actual data in the log.

In the OData API Audit Log, you will see the the filter bar is displayed horizontally above the grid table and has the following default filter items – *Session ID*, *Request ID*, *Status*, *OData API Call*, *Entity Name*, and *Request Time (From)*.

The audit log comprises the following information:

- *Log ID* (filterable)
- *Login ID* (filterable)
- *Session ID*
- *Request ID*
- *Status*
- *Request Time*
- *OData API Call*
- *Entity*
- *Responses (ms)* (filterable)
  The Response filter supports >, <, >=, <=, != and the filter 100..200 which means >=100 and <=200.
- *HTTP Message* (audit log detail - this can be downloaded)

6.4.3 SF API Audit Log

Describes the features, and required permissions for the SF API Audit Log.

In the SF API Audit Log, you can display audit log information such as the time a request was made, the time taken for the response, the type of SF API call, and so on.

The look and feel of the OData API audit log has been enhanced. It's available in the Admin Center. If you can't see it there, check that you have the permission, *Access to SF API Audit Log* (available in *Manage Integration Tools* and applicable to both RBP and non-RBP users).

Because the Audit Log may contain sensitive data, ensure that access to the Audit Log is given only to trusted users. One safety strategy would be to grant permission to developers only when necessary for development and debugging purposes, and then remove access to this feature.

Note

Data is captured in the Audit Log whether any user has permission to view the log or not. Therefore, removing a user's access to the log will not remove the actual data in the log.

In the SF API Audit Log, you will see the the filter bar is displayed horizontally above the grid table and has the following default filter items – *Session ID* (uses Contain operator), *Correlation ID* (uses Equal operator), *Status*, *SFAPI API Call*, *Object*, and *Request Time (From)*, and *Request Time (To)*.

The audit log comprises the following information:

- *Log ID* (filterable)
- *Login ID* (filterable)
- *Session ID*
- *Correlation ID*
- *Status*
- *Request Time*
- *SFAPI Call*
- *Responses (ms)* (filterable)
  The Response filter supports >, <, >=, <=, != and the filter 100..200 which means >=100 and <=200.
- *HTTP Message* (audit log detail - this can be downloaded)

6.4.4 Manage User API Options

To manage optional actions that occur when user write operation is performed, you can use this feature. Please note that in the Admin Center, this is called *Manage API Option Profile* but in the API Center, it is called *Manage User API Options*.

Here are some of the features available to you for managing user API options. In your instance you'll find more features and information in the in-app help:

- Send a welcome e-mail to all new users
- Determine the default password for new users - for example, you can choose to use the user name, user ID, e-mail address, or, a randomly generated system password.
- Determine how forms are routed - for example one of options available is *Automatic En Route Document Transfer to New Manager*
- Determine how automatic document removal is performed - for example one of available options is *Remove Inactive Employees In-Progress Documents*.

6.4.5 OData IP Whitelisting

To configure IP whitelisting when basic authentication is used, you can use this feature. Please note that this feature is called *OData API Basic Authentication Configuration* in the Admin Center but in the API Center it is called*OData IP Whitelisting*.

You have the following options for enforcing basic authentication:

- Always
- Never
- Use basic authentication for specified IP addresses

6.4.6 Legacy SFAPI IP Whitelisting

https://jira.successfactors.com/issues/?jql=(filter%20in%20(59143)%20AND%20filter%20in%20(50469)%20OR%20key%20%3D%20TLS-6659)%20AND%20project%20%3D%20API

6.5 Refreshing and Exporting the Metadata

To view OData API Metadata Refresh and Export, select the *OData API Metadata Refresh and Export* link. You will see two buttons:

- *Refresh*
- *Export*

Refresh and Export operations are described below:

Operation	Description
Refresh	Refreshes the metadata cache. The updated metadata is re-generated, and saved into the cache and is available for future operations such as export.
Export	Exports and downloads the metadata. The default export file name is `CompanyName-Metadata.xml`

In most cases, refresh is automatic, for example when an Admin changes or creates an object definition in the Extension Center or in the Configure Object Definition page or when an Admin enables/disables a feature in the Upgrade Center leading to a change that requires a refresh of OData metadata.

Use the Refresh option whenever entities or fields are not reflecting the latest changes made to them.

Note

A manual refresh is always required when a user with provisioning access chooses to enable/disable a product feature or change the language packs via provisioning leading to a corresponding OData API change. In this case, the change will only be effective after a manual refresh.

7 OData Service Catalog

Service Catalogs provide a way to group APIs together in contextually-relevant groups, greatly reducing the size of the $metadata for each catalog.

Use the `http://host/odata/v2/CatalogService/$metadata` API call to generate a catalog of services.

Sample XML Response

Sample XML response is shown below :

```
<?xml version='1.0' encoding='utf-8'?>
  <edmx:Edmx Version="1.0" xmlns:edmx= "http://schemas.microsoft.com/ado/2007/06/
edmx"  xmlns:atom= "http://www.w3.org/2005/Atom" >
    <edmx:DataServices m:DataServiceVersion= "2.0" xmlns:m= "http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata" >
      <Schema Namespace= "CATALOGSERVICE" xmlns= "http://schemas.microsoft.com/ado/
2008/09/edm"  xmlns:sf= "http://www.successfactors.com/edm/sf" >
        <EntityType Name= "Service">
          <Key>
            <PropertyRef Name= "ID" />
          </Key>
          <Property Name= "ID" Type= "Edm.String " Nullable= "false " />
          <Property Name= "Description" Type= "Edm.String " Nullable= "false " />
          <Property Name= "Author" Type= "Edm.String " Nullable= "false " />
          <Property Name= "MetadataUrl" Type= "Edm.String " Nullable= "false " />
          <Property Name= "ServiceUrl" Type= "Edm.String " Nullable= "false " />
        </EntityType>
        <EntityContainer Name= "CATALOGSERVICE_Entities"
m:IsDefaultEntityContainer= "true ">
          <EntitySet Name= "ServiceCollection" EntityType=
"CATALOGSERVICE.Service" />
        </EntityContainer>
        <atom:link rel= "self" href= "http://performancemanager4.successfactors.com:
8020/odata/v2/CatalogService/$metadata"  />
        <atom:link rel= "latest-version" href= "http://
performancemanager4.successfactors.com:8020/odata/v2/CatalogService/$metadata"  />
        <atom:link rel= "analytics" href= "https://analytics4.successfactors.com/
ProductionData/ODataService.svc"  />
      </Schema>
    </edmx:DataServices>
</edmx:Edmx>
```

Response Description

This XML output follows regular OData metadata format. It contains 2 parts:

1. An OData entity named 'Service'. Currently this entity is not accessible excepts it's definition in metadata.

2. Several `<atom:link>` links, each of which publishes a service URL. Each data center is be configured to publish different service URLs. The `self` and `latest-version` links always point to the catalog service itself. The `analytics` link published the Analytics OData API URL which is the only URL available currently.

8 OData Metadata Document

This section describes the SuccessFactors OData document.

The metadata document is a static resource that describes the data model and type system understood by that particular OData service. You can use the metadata document to learn how to query and navigate between the entities in the system. Metadata extensions provide additional metadata information on the top of AtomPub, which gives you access to advanced operations such as data retrieval filtration. To access the SuccessFactors HCM Suite OData metadata document, go to https://<hostname>/odata/v2/$metadata. This call supports locale based labels. The default label is "en_US".

The URL will return an XML serialization of the service, including the Entity data model and the service operation descriptions. The metadata response supports only `application/atom+xml` type. (For example, the metadata response cannot be accessed in JSON).

Cache-Control for OData API $metadata Operations

The value of `ETag` is used to check if the metadata saved in client cache is the same as the one on the server. The value of max-age in the Cache-Control header is set to the life-cycle of the client metadata cache. If the cache is valid, no new request is sent to the server. When a client raises a request for metadata the first time, the server sends back a response with the latest metadata, along with a response header named ETag. The value of ETag is unique and matches the metadata version. This value is used for “If-None-Match” in the request header the next time the same request is raised. The server checks the “If-None-Match” value when a new request arrives. If its value is the same as the latest ETag generated by the server, the server simply sends back a status code of 304 (Not-Modified) instead of resending the entire metadata.

Metadata document and the API

The metadata document describes the capabilities of the API to your SuccessFactors HCM Suite instance. It contains the details of each Entity that is accessible through the API, including fields, their names and labels, their data types, and the relationships (associations) between the Entities.

The metadata document also describes the operations available in the API. The OData protocol specifies four basic data base style operations: Insert, Update, Query and Delete. SuccessFactors has added a fifth operation called “Upsert” which performs an “Insert or Update” operations. In the future SuccessFactors may add many other custom operations. Typically custom operations will be perform specific business transactions, especially if a custom API is easier to manage versus a data base style approach against multiple entities.

Development using the metadata document and API

Regardless of which operations are used (create, read, update, delete, upsert, or even custom operations), SuccessFactors HCM Suite will apply the appropriate business logic for each Entity. In other words, even though the operations appear to be database centric, the API goes through the application business logic layer. The API does not go directly against the database, nor does it bypass the business logic layer. The Entities in the API represent logical application objects familiar to an application user. Note that the Entities do not represent the actual physical data storage implementation, which may be in a different structure.

Using the metadata document for customized development is optional and considered advanced behavior for API clients. It can be critical to API client systems that need to write general code which automatically adjusts to the system configuration. For example, if you are writing a middleware tool that allows runtime discovery of the SuccessFactors HCM OData system, you can use the metadata document to discover the Entities and fields, and their data types.

8.1 EntitySet

The OData Metadata document for SuccessFactors HCM Suite contains the following information about the EntitySet extension.

The EntitySet SF Extension

ATTRIBUTE	DEFAULT	DESCRIPTION
sap: creatable	`True`	Instances of this Entity type can be created.
sap: updatable	`True`	Instances of this Entity type can be updated.
sap :upsertable	`True`	Instances of this Entity type can be upserted.
sap: deletable	True	Instances of this entity can be deleted.

Example

Sample Code

```
<EntitySet Name="PicklistOption" EntityType="SFOData.PicklistOption"
sap:label="PicklistOption" sap:creatable="true" sap:updatable="true"
sap:upsertable="false" sap:deletable="false">
```

8.2 AssociationSet

The OData Metadata document for SuccessFactors HCM Suite contains the following information about AssociationSet.

The AssociationSet SF Extension

ATTRIBUTE	REQUIRED	DEFAULT	DESCRIPTION
`sap:creatable`	No	`True`	Instances of this relation can be created.
sap:updatable	No	`True`	Instances of this relation can be updated.
sap:upsertable	No	`True`	Instances of this relation can be upserted.

Example

```
<AssociationSet Name="picklistoption_rel_picklistlabel"
    sap:insertable ="true" sap:updatable="true" sap:upsertable="true"
    Association="SFOData.picklistoption_rel_picklistlabel">
  <End EntitySet="PicklistOption" Role="picklistoption" />
  <End EntitySet="PicklistLabel" Role="picklistlabel" />
</AssociationSet>
```

8.3 Property (Deprecated and Removed)

These attributes have been deprecated since 1511 and have now been removed (1608). This is for information only.

The Property SF Extension

ATTRIBUTE	REQUIRED	DEFAULT	DESCRIPTION	COMMENTS
sf:Insertable	No	`True`	Value of the property can be set by client during insertion; "`false` if the value is generated by server.	This is equivalent to `sap:creatable`.
sf:Updatable	No	`True`	Value of the property can be updated..	This is equivalent to `sap:updatable`.
sf:Upsertable	No	`True`	Value of the property can be upserted.	
sf:Required	No	`False`	Indicate the property is required during insertion.	

ATTRIBUTE	REQUIRED	DEFAULT	DESCRIPTION	COMMENTS
sf:Selectable	No	`True`	Indicate the property can be used in `$select.`	
sf:Sortable	No	`False`	Indicate the property can be used in `$orderby`.	This is equivalent to `sap:sortable`.

Example

```
<Property Name="status" Type="Edm.String" Nullable="true"
    sf:Insertable="true" sf:Updatable="true" sf:Upsertable="true"
    sf:Selectable="true" sf:Sortable="true" sf:Filterable="true"
    MaxLength="9">
```

8.4 NavigationProperty (Deprecated)

The OData Metadata document for SuccessFactors HCM Suite contains the following information about the NavigationProperty extension.

Note

These attributes have been deprecated since 1511 and removed since (1608). This is for information only.

The NavigationProperty SF Extension

ATTRIBUTE	REQUIRED	DEFAULT	DESCRIPTION	COMMENTS
sf:Insertable	No	`True`	Reference of the navigation property can be set by client during insertion; `false` if the value is generated by server.	
sf:Updatable	No	`True`	Reference of the navigation property can be updated.	
sf:Upsertable	No	`True`	Reference of the navigation property can be upserted .	
sf:Required	No	`False`	Indicate that the property is required during insertion	
sf:Selectable	No	`True`	Indicate that the property can be used in `$select`.	

ATTRIBUTE	REQUIRED	DEFAULT	DESCRIPTION	COMMENTS
sf:Sortable	No	`False`	Indicate the property can be used in `$orderby`.	
sf:Filterable	No	`False`	Indicate the property can be used in `$filter`.	

Example

```
<NavigationProperty Name="parentPicklistOption"
    sf:Insertable="true" sf:Updatable="true" sf:Upsertable="true"
    sf:Selectable="true" sf:Sortable="true" sf:Filterable="true"
    sf:InlineInsertable="false" sf:InlineUpsertable="false"
    Relationship="SFOData.picklistoption_rel_parentpicklistoption"
    FromRole="picklistoption" ToRole="parentpicklistoption" />
```

8.5 Metadata Enhancements for UI5 Smart Controls

SAP extensions and their properties for U15 smart controls are described here.

For more information about Extensions Specification in OData, please visit https://scn.sap.com/docs/DOC-44986.

Entity Type

Name	Required	Default	Comment
sap:content-version	No		
sap:semantics	No		values can be these: • vcard • vevent • vtodo • parameters • aggregate • variant
sap:label	No		
sap:is-thing-type	No	false	

Properties

Navigation Properties

Name	Display by Default	Default Value	Equivalent to	Comment
sap:creatable	No	true	sf:Insertable	Reference of the navigation property can be set by client during insertion; “false” if the value is generated by server
sap:updatable	No	true	sf:Updatable	
sap:upsertable	No	true	sf:Upsertable	
sap:required	No	false	sf:Required	Indicates that the property is required during insertion
sap:visible	No	true	sf:Selectable	Indicates that the property can be used in $select
sap:sortable	No	false	sf:Sortable	
sap:filterable	No	false	sf:Filterable	Indicates that the property can be used in $filter
sap:picklist	No	null	sf:Picklist	

Function Imports

Name	Display by Default	Default Value	Comment
sap:action-for	No		Value must be the name of an existing entity, such as ‘User’.
sap:label	No		

Name	Display by Default	Default Value	Comment
sap:applicable-path	No		Value must be of a name of boolean type property of the entity that action-for describes, such as 'male',which is a property of 'User'; or a name like 'A/B', A is a complex type property of the entity that action-for describes and B is a boolean type property of A. *The property that applicable-path describes have dependency on the entity that action-for describes.*

Schema

Name	Display by Default	Default Value	Comment
sap:content-version	No		

EntitySet

Name	Display by Default	Default Value	Comment
sap:content-version	No		
sap:creatable	No	true	
sap:updatable	No	true	
	No	true	
sap:deletable	No	true	
sap:searchable	No	false	
sap:pageable	No	true	
sap:topable	No	true	

Name	Display by Default	Default Value	Comment
sap:subscribable	No	false	
sap:addressable	No	true	
sap:requires-filter	No	false	
sap:label	No		
sap:semantics	No		

AssociationSet

Name	Required	Default Value	Comment
sap:content-version	No		
sap:semantics	No		
sap:creatable	No	true	
sap:updatable	No	true	
	No	true	
sap:deletable	No	true	

Association

Name	Required	Default Value	Comment
sap:content-version	No		
sap:semantics	No		

Parameter

Name	Required	Default Value	Comment
sap:label	No		

Annotations (SAP Vocabulary)

Name	Required	Default Value	Comment
ValueList	No		com.sap.vocabularies.Common.v1.ValueList
LineItem	No		com.sap.vocabularies.UI.v1.LineItem

8.6 Retrieving metadata

In order to get extra information, such as language labels, picklists, beyond what the standard OData metadata provides, Successfactors OData exposes metadata as an entity. Take a look a thte sample API calls below to make the most of this feature.

Sample API Calls

The following API call shows you how to display a list of entities in your instance:

```
https://<hostname>/odata/v2
```

The following API call shows you how to access the entire metadata for your instance:

```
https://<hostname>/odata/v2/$metadata
```

The following API call shows you how to access the metadata for only the User entity:

```
https://<hostname>/odata/v2/Entity('User')?$format=json
```

Here the entity properties are exposed as a complex type value embedded in the response body of 'Entity'. Different forms of metadata can thus be exposed without changing the standard OData metadata format. You can access the new metadata just like you would access a regular entity. In addition, it supports a simple filter to output metadata of a specific entity or a group of entitie

The following API call shows you how to access the metadata for the User and Photo entities:

```
https://<hostname>/odata/v2/User,Photo/$metadata
```

You can use the same URL to query User entity data:

```
https://<hostname>/odata/v2/User,Photo/User?$format=json&$filter=userId eq ‘cgrant’
```

9 Appendix of Error Messages

This section contains description of error messages specific to the SuccessFactors HCM Suite.

9.1 OAuth 2.0 Related Error Messages

Error Messages List for OAuth 2.0

ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_INVALID_CLIENT_INFO	500	Client information contains error.	
OAUTH2_ERROR_MISSING_REQUIRED_CLIENT_INFO_FIELD	500	Missing required field in client information.	
OAUTH2_ERROR_INVALID_CLIENT_INFO_FIELD	500	Some client information field contains an error.	
OAUTH2_ERROR_CLIENT_ALREADY_EXIST	500	The client application (API key = "###") already exists.	
OAUTH2_ERROR_CLIENT_NOT_EXIST	500	The client application (API key = "###") does not exist.	
OAUTH2_ERROR_UNABLE_TO_REGISTER_CLIENT	500	Uable to register client application.	
OAUTH2_ERROR_UNABLE_TO_DEREGISTER_CLIENT	500	Unable to deregister client application.	
OAUTH2_ERROR_INVALID_TOKEN_URL	500	The token URL is invalid.	The token URL is in an incorrect format.
OAUTH2_ERROR_INVALID_X509_PRIVATE_KEY	500	The X.509 private key contains an error.	The X.509 private key is invalid.
OAUTH2_ERROR_UNABLE_TO_GENERATE_SAML_ASSERTION	500	Unable to generate SAML assertion.	OAuth server is unable to construct a valid X.509 private key object.
OAUTH2_ERROR_UNABLE_TO_SIGN_SAML_ASSERTION	500	Unable to sign SAML assertion.	Failed to sign SAML assertion using the provided X.509 private key.

ERROR STRING	HTTP RE-SPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_MISSING_REQUIRED_PARAM	400	Parameter "###" is required in the OAuth request.	For a `get` access token request, `grant_type`, `client_id`, `company_id` and `assertion` fields are required. To generate a SAML assertion, `client_id`, `token_url`, `user_id` and `private_key` fields are required.
OAUTH2_ERROR_INVALID_GRANT_TYPE	400	The grant_type is not supported.	`grant_type` must be set to `urn:ietf:params:oauth:grant-type:saml2-bearer`
OAUTH2_ERROR_UNABLE_TO_COL-LECT_SAML_ASSERTION	400	Unable to retrieve SAML assertion.	The SAML assertion is not Base64 encoded. Missing `Issuer`, `SubjectNameId`, `Audiences`, or `Recipients` in the assertion.
OAUTH2_ERROR_SAML_ASSERTION_EXPIRED	400	The provided SAML assertion is expired.	The SAML assertion usually expires in 5-10 minutes.
OAUTH2_ERROR_UNABLE_TO_VERIFY_SAML_AS-SERTION	401	Unable to verify the signature of the SAML assertion.	SAML assertion is verified using the same X. 509 certificate that was provided during OAuth client application registration.
OAUTH2_ERROR_UNABLE_TO_VALI-DATE_SAML_ASSERTION	401	Unable to validate "###" in the SAML assertion.	The `Issuer` field in the assertion does not match the API_KEY. `Audiences` does not include www.successfactors.com. `Recipients` points to a different token host.
OAUTH2_ERROR_COMPANY_NOT_EXIST	401	Company "###" does not exist.	
OAUTH2_ERROR_COMPANY_INACTIVE	401	Company "###" is not currently active.	
OAUTH2_ERROR_COMPANY_LICENSE_EXPIRED	401	Company "###" has expired license.	

ERROR STRING	HTTP RE-SPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_API_KEY_NOT_EXIST	401	API key ###"" does not exist in company "###" .	
OAUTH2_ERROR_API_KEY_DISABLED	401	API key "###" has been disabled in company "###" .	
OAUTH2_ERROR_AUTHENTICATION_FAILED	401	Unable to authenticate the client.	An OAuth server error occurred while authenticating `access_token`.
OAUTH2_ERROR_AUTHORIZATION_FAILED	403	Unable to authorize the client.	An OAuth server error occurred while authorizing `access_token`.
OAUTH2_ERROR_UNABLE_TO_SIGN_TOKEN	403	Unable to sign access token.	The OAuth server uses a SuccessFactors server certificate to sign the access token. An error occurs if the access token has already been signed, or if the SuccessFactors server certificate is missing.
OAUTH2_ERROR_UNABLE_TO_GRANT_TOKEN	403	Unable to grant access token.	Content is missing in the access token, or a runtime error happened while constructing the access token (this includes JSON-generation exceptions).
OAUTH2_ERROR_UNABLE_TO_BUILD_TOKEN_RESPONSE	500	Unable to build OAuth response.	
OAUTH2_ERROR_MISSING_REQUIRED_HEADER	400	Header "###" is required.	Missing `Authorization` header in validate token request, or when protected resources were accessed.
OAUTH2_ERROR_MISSING_ACCESS_TOKEN	400	Access token is required.	`Authorization` header is empty.
OAUTH2_ERROR_MUST_BE_BEARER_TOKEN	400	Access token has to be a Bearer token.	`Authorization` header is not a `Bearer` format.
OAUTH2_ERROR_UNABLE_TO_COLLECT_TOKEN	400	Unable to retrieve access token.	The access token is missingcontent, or a parsing error happened while deserializing an access token in JSON format.

ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_UNABLE_TO_VERIFY_TOKEN	401	Unable to verify the signature of the access token.	The access token has modified the HTTP MITMA. The SuccessFactors server public key couldn't be found.
OAUTH2_ERROR_UNABLE_TO_VALIDATE_TOKEN	401	Unable to validate access token.	The Access token contains invalid content or has insufficient information (for example, missing `client_id`, `company_id`, and so on).
OAUTH2_ERROR_TOKEN_INVALID_APIKEY	401	The access token does not apply to API key "###" .	
OAUTH2_ERROR_TOKEN_INVALID_COMPANY	401	The access token does not apply to company "###" .	
OAUTH2_ERROR_TOKEN_INVALID_APPLICATION_NAME	401	The access token does not apply to application name "###".	Verify the application name specified during the OAuth client application registration.
OAUTH2_ERROR_TOKEN_REJECTED_OR_EXPIRED	403	The access token is either rejected or expired.	Token has expired (default is 24 hours).
OAUTH2_ERROR_INSUFFICIENT_PERMISSION_SCOPE	403	Insufficient permission scope.	
OAUTH2_ERROR_UNABLE_TO_GRANT_ACCESS	403	Unable to grant access.	External User.
OAUTH2_ERROR_UNABLE_TO_BUILD_RESPONSE	500	Unable to build token validation response.	
INTERNAL_ERROR	500	Internal error.	

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: https://help.sap.com/viewer/disclaimer).

go.sap.com/registration/contact.html